Statistical Tools to Model Space-Time Data with a Focus on Biodiversity Applications

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Florian Daniel Markus Gerber

aus

Langnau im Emmental BE

Promotionskommission

Prof. Dr. Reinhard Furrer (Vorsitz)
Dr. Gabriela Schaepman-Strub
Prof. Dr. Michael E. Schaepman
Prof. Dr. Torsten Hothorn

Zürich, 2017

Zusammenfassung

Statistische Modelle sind wichtige Hilfsmittel um Raum-Zeit-Daten wie Satellitenbilder und ökologische Feldmessungen zu analysieren und interpretieren. Dabei verunmöglichen komplexe Datenstrukturen und immer grössere Datenmengen den Gebrauch von herkömmlichen geostatistischen Methoden wie Kriging. Diese Unzulänglichkeit eröffnet das aktive und attraktive Forschungsgebiet der angewandten Raum-Zeit-Statistik für grosse Daten. Die in dieser Arbeit präsentierten Fortschritte auf diesem Gebiet sind hauptsächlich durch ökologische Fragestellungen betreffend die arktische Vegetation und deren Anpassungen an die globale Klimaerwärmung motiviert. Quantitative Aussagen über die arktische Vegetation beruhen hauptsächlich auf zwei fundamental verschiedenen Arten von Messungen: Die eine Art besteht aus Feldmessungen von biologisch relevanten Parametern, die andere stützt sich auf Fernerkundungsdaten und die daraus abgeleiteten Vegetationsindizes. Beide Ansätze führen zu Raum-Zeit-Daten und bringen verschiedene Probleme mit sich, welche gültige Aussagen für die ganze Arktis erschweren. Zum Beispiel gibt es relativ wenige Orte mit Feldmessungen und die Fernerkundungsdaten sind häufig beeinträchtigt durch mit Wolken, Schnee und Wasser bedeckte Landschaften. Diese Doktorarbeit präsentiert eine Reihe von statistischen und rechnerischen Entwicklungen, welche helfen die Aussagen zur Vegetation der Arktis zu präzisieren.

Die Arbeit ist in fünf Manuskripte aufgeteilt: **Paper I** behandelt den 64-bit Ausbau der R Erweiterung *spam*, welche neu dünnbesetzte Matrizen mit mehr als 2^{31} von Null verschiede Einträgen manipulieren kann. Besagter Ausbau ermöglichte grosse fernerkundungsbasierte Vegetationsindex Daten mit einem nicht stationären Gauss-Prozess zu modellieren. Die 64-bit Erweiterung basiert auf der R Erweiterung *dotCall64*, welche in **Paper II** detailliert diskutiert wird. Ferner beschreibt **Paper III** eine neue Methode um fehlende Werte in raum-zeitlichen Fernerkundungsdaten zu berechnen. Dabei berechnet die Methode jeden fehlenden Wert einzeln. Sie sucht eine geeignete Raum-Zeit-Teilmenge der Daten und wendet Sortieralgorithmen für Bilder sowie Quantilsregression an. Um auch sehr grosse Daten mit leistungsstarken Rechnern bearbeiten zu können verfügt die dazugehörige R Erweiterung *gapfill* über ein modulares Design mit Möglichkeiten zur parallelen Datenverarbeitung. **Paper IV** behandelt verschiedene Umsetzungs- und Validationsstrategien von bayesschen hierarchischen Modellen für Zähldaten. Wie in der Einleitung dieser Arbeit skizziert sind Fortschritte auf diesem Gebiet vielversprechend um Daten von verschiedenen Quellen, zum Beispiel Daten zum Vorkommen von Pflanzenarten und Vegetationsindex Daten, gemeinsam zu modellieren. Schliesslich stellt **Paper V** eine Fallstudie vor, welche arktische Feldmessungen der Biodiversität mit einer fernerkundungsbasierten Landschaftscharakterisierung verbindet. Genauer werden die Abhängigkeiten zwischen Biodiversitätsindizes basierend auf Daten des Arctic Vegetation Archive und Landschaftscharakterisierungen mit Vegetationsindex Daten und einem Höhenmodell untersucht.

Abstract

Statistical models are important means to analyze and interpret space-time data, such as satellite datasets and ecological field measurements. However, complex data structures and increasing dataset sizes make it impossible to use standard geostatistical methods like kriging. The resulting methodological gap opens up an active and attractive research area, namely the one of applied spatio-temporal statistics for large datasets. The herein presented advances in that field are mainly motivated by ecological research questions centered around the Arctic vegetation and its response to global warming. Quantitative statements about the Arctic vegetation are typically based on two fundamentally different types of measurements: field measurements of biologically relevant parameters on the one hand and remotely sensed vegetation indices on the other. Both techniques lead to spatio-temporal data and face various challenges, which make it difficult to characterize vegetation at Pan-Arctic scale. For instance, the spatial sparsity of field measurements and the fact that satellite observations are often confounded by cloud, snow, and water covered surfaces are major drawbacks. This PhD thesis presents a series of statistical and computational developments, which help to improve the quality of quantitative statements about the Arctic vegetation.

The thesis is structured into five self-contained paper manuscripts: **Paper I** is concerned with making the sparse matrix algebra R package *spam* capable of handling large 64-bit matrices with 2^{31} and more non-zero elements. This enabled fitting a non-stationary spatial Gaussian process model to a large remote sensing based vegetation index dataset. The 64-bit extension is based on the R package *dotCall64*, which is discussed in detail in **Paper II**. **Paper III** introduces a new spatio-temporal prediction method for missing values in satellite data. The method predicts each missing value separately by selecting a suitable spatio-temporal subset followed by an image sorting procedure and quantile regression. To be able to process massive amounts of data with large computer systems the corresponding R package *gapfill* features a modular design with an emphasis on parallel computing. **Paper IV** elaborates on different implementation and validation strategies for spatial Bayesian hierarchical models for count data. As sketched in the introduction of the thesis, advances in that direction are promising to jointly model data from various sources, such as Arctic plant abundance data and remotely sensed vegetation indices. Eventually, **Paper V** presents a case-study, in which Arctic plot scale biodiversity measurements are related to remote sensing based landscape characterizations. More precisely, relations between biodiversity indices derived from field measurements of the Arctic Vegetation Archive and landscape characterizations based on vegetation index data as well as a digital elevation model are explored.

Contents

Preface

Statistical models are an important mean to draw scientific conclusions from observations. Based on a set of assumptions they summarize data and quantify uncertainties. In particular, the uncertainty statements are essential to measure the information content of collected data and to make data driven decisions. The research area concerned with developing statistical methods to answer questions from research fields outside statistics is termed "applied statistics." As scientific hypothesis and data generating processes are evolving, there is a need to refine and further develop the statistical methods used to analyze them. For example, the increasing dataset sizes generated by satellite sensors and other high-resolution measurement devices motivate the recent efforts to extend spatio-temporal models to large datasets. The research presented in this thesis is concerned with applied spatio-temporal statistics and is motivated by the ecological study area outlined next.

Climate warming is particularly pronounced in the Arctic region and triggers changes in the ecological system of the tundra. Investigating such changes not only leads to a better understanding of the Arctic flora, but also helps to reduce a major source of uncertainty in the predictions of future climatic conditions on Earth. There are two fundamentally different approaches to quantify changes in the Arctic vegetation: One approach is based on field measurements of biologically relevant parameters and typically leads to precise and localized descriptions. As collecting such data is expensive, they are sparse in space and time, i. e., the amount of not measured areas is much larger compared to the total area of interest. Thus, the generalization of findings with respect to the entire Arctic region is delicate. The other approach relies on remotely sensed measurements, which are used to characterize vegetation via vegetation indices, such as the normalized difference vegetation index (NDVI). While such indices are available for relevant areas and time periods, their explanatory power is limited by a relatively low spatial resolution and the corruptions caused by clouds, snow, and other factors. Evidently, a more accurate description of the ecological system could be obtained via the combination of information from both approaches. The role of applied statistics in this context is to develop methods that retrieve the relevant information from the available data and to quantify uncertainty.

Analyzing Arctic field measurements and vegetation index data is a challenging task because the data involved have various data structures and suffer from difficult data collection conditions. This thesis presents further developments of statistical methods that contribute to a better understanding of the Arctic vegetation. The main advanced made with that respect are presented in the five self-contained papers listed on page 5; see the outline given on page 39 for more detailed information on their contents and authors contributions also. Paper I (p. 45) and paper II (p. 59) introduce a strategy that allows passing large 64-bit vectors from the R to compiled C/C++ or Fortran code. While Paper I illustrates the new capabilities with a non-stationary spatial model fitted to a large remote sensing based NDVI dataset, Paper II rather gives technical insights into the software implementation. In Paper III (p. 79) we present a new spatio-temporal method to predict missing values in remote sensing based vegetation indices. The method is tested with

NDVI data from the Arctic region and compared to state-of-the-art methods. Paper VI (p. 97) elaborates on different implementation and validation strategies of spatial Bayesian hierarchical models for count data. Advances in that direction are promising to jointly model, for example, plant abundance data with NDVI observations. Paper V (p. 131) presents a case-study, in which Arctic plot scale biodiversity measurements are related to remote sensing based landscape characterizations. Relevant background information on the ecological and statistical topics is given in the introduction on page 9.

The here presented work was carried out in the group of Prof. Dr. R. Furrer at the Department of Mathematics of the University of Zurich (UZH) between August 2013 and July 2017. Dr. G. Schaepman-Strub from the Department of Evolutionary Biology and Environmental Studies UZH co-supervised the project. Prof. Dr. T. Hothorn from the Epidemiology, Biostatistics and Prevention Institute UZH and Prof. Dr. M. E. Schaepman head of the Remonte Sensing Laboratories UZH are members of the PhD committee. This PhD project was part of and funded by the University Research Priority Program on Global Change Biodiversity (URPP GCB) launched by the UZH in 2013. The URPP GCB aims at studying changes in ecosystem processes and services that are essential to human well-being. It focuses on biodiversity as a key concept to describe ecosystems and their interactions with a changing global environment. Thereby, biodiversity is both a response variable reacting to environmental changes and a factor modifying ecosystem processes. To investigate these feedback mechanisms at different scales the URPP GCB follows an interdisciplinary approach and gathers researchers with expertise in remote sensing, ecosystem functioning research, soil science, evolution, human actors, environmental justice, physics, and statistics. From an organization point of view, the URPP GCB is divided into eight interdisciplinary projects. The presented work contributes to Project 7 "Ecosystem-environment models: linking sparse and local field observations with extensive but low-resolution satellite data."

Introduction

The statistical and computational developments presented in this thesis are motivated by ecological research questions related to vegetation changes in the Arctic tundra. In order to better understand the connection between statistics and the Arctic ecology, this introduction provides a brief overview of relevant methods and findings of both research areas. Section 1 covers the Arctic vegetation and its interactions with climate change. Besides the current state of research also opportunities are highlighted that improve the understanding of the Arctic ecosystem with further developments of statistical tools. Section 2 introduces relevant statistical concepts with an emphasis on Bayesian hierarchical models for data fusion.

1 Ecological motivation: Understanding vegetation changes in the Arctic

Global climate change affects life on Earth in many respects and therefore constitutes an important field of research. A scientific view on this highly interdisciplinary subject is given by the Intergovernmental Panel on Climate Change (IPCC, www.ipcc.ch). In their Fifth Assessment Report (IPCC, 2013) the researchers present methods to assess global change, quantify observed changes, and predict future climate. One of their findings is that some consequence of global warming, such as the observed increases of global average temperature, changes in precipitation patterns, and sea-level rise, are already observed today (Karl and Trenberth, 2003; Rahmstorf, 2007). Thereby not only average values are affected, but also the frequencies of extreme events, which is of major importance to life on Earth. Moreover, there is broad agreement on anthropogenic carbon dioxide emissions being the major driver of global warming.

In order to characterize and predict climate change a fundamental understanding of climate relevant processes is necessary. As the Earth is a dynamic system with many temperature dependent processes, global warming triggers a series of changes, which, in turn, influence global warming itself. An example of such a feedback mechanism is the reduced sea ice cover, which is caused by global warming and alters the absorption properties of the Earths surface such that it takes up more energy from solar radiation (Deser et al., 2000). While quantifying feedback loops is challenging and requires the development of physical and biological models, they are essential to understand climate change. This introductory section is concerned with the Arctic ecosystem in view of global warming. More precisely, we focus on the vegetation of Arctic tundra, which is of great interest to climate change research as it is strongly affected by climate change and, at the same time, has a great potential to feedback to global warming.

1.1 Characterize vegetation with data

Field measurements

Field measurements lead to a detailed and local characterization of vegetation. Commonly measured parameters of Arctic vegetation are canopy height, biomass, and species abundance information (e. g., see the raw data of Elmendorf et al., 2012a). As taking such measurements is time-consuming, it is not possible to characterize a landscape by measuring the entire vegetation in its area. Instead, vegetation is measured at sample plots only. To characterize larger areas sample designs are helpful, which typically lead to hierarchical data structures. Several plot measurements are combined to describe one type of landscape. The actual measurements at the sample plots can be taken by one of the methods discussed by Mamet et al. (2016). Often the resulting measurements are stored together with meta-data describing the geographical position and date of each record. Thus, plot data have both a spatial and a temporal component, and are often of an irregular structure (i. e., the spatial and temporal distances between observations are irregular). Specialized software and database systems help to manage such datasets (e. g., Hennekens and Schaminée, 2001). Moreover, ongoing efforts aim at harmonizing data collection methods and making them publicly available; see International Tundra Experiment (ITEX, http://ibis.geog.ubc.ca/itex) and the Arctic Vegetation Archive (AVA, www.geobotany.uaf.edu/ava).

Remotely sensed measurements

Optical remote sensing is used to monitor Earth surface processes (Schaepman-Strub et al., 2006, 2009). It can be used to quantify vegetation in terms of vegetation indices, which are designed to identify vegetation based on the characteristic reflectance spectra of green leafs; see de Jong (2012) and Jones and Vaughan (2010) for introductions. One of the most prominent vegetation index is the normalized difference vegetation index (NDVI) defined as

$$\mathrm{NDVI} = \frac{\rho_{\mathrm{NIR}} - \rho_{\mathrm{red}}}{\rho_{\mathrm{NIR}} + \rho_{\mathrm{red}}} \in [-1, 1],$$

where ρ_{NIR} and ρ_{red} are reflectance factors of the near infrared and red spectra, respectively. Besides the sensitivity of the NDVI to green vegetation only, it is also affected by other ground-cover characteristics, such as the type of vegetation, litter, bare ground and soil-moisture making its interpretation difficult (Tucker et al., 2005). Several satellite missions provide preprocessed datasets that are suitable to derive vegetation indices. For example, the global MOD13 data product from the moderate-resolution imaging spectroradiometer (MODIS) features an NDVI data layer with a spatial resolution of 500 m at bimonthly time intervals (Huete et al., 2002; Didan et al., 2015). Another example is the NDVI_{3g} product from the advanced very-high-resolution radiometer (AVHRR) sensors, which has a spatial resolution of 8 km and bimonthly records going back until 1981 (Pinzon and Tucker, 2014). Both data products have regularly spaced observations in space and time, and hence, can be stored in array like data structures.

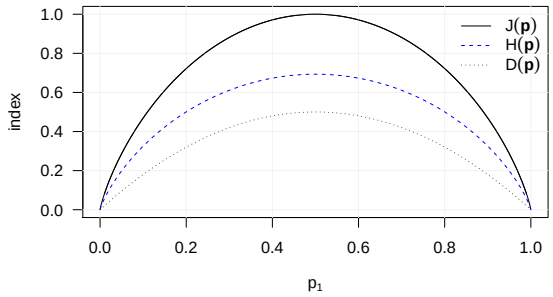


Figure 1: *The evenness $J(\mathbf{P})$, Shannon $H(\boldsymbol{p})$, and Simpson $D(\boldsymbol{p})$ diversity indices in a setting with two species ($S = 2$). The indices are plotted as a function of p_1 being the proportion of species one.*

Optical satellite observations are sensitive to factors such as atmospheric conditions and angular configurations. As a consequence, NDVI data products typically exhibit a considerable amount of low-quality values, which hinder a proper parameterization of continuous vegetation change (Stow et al., 2004; Hmimina et al., 2013). Many procedures have been proposed to reconstruct complete data products based on the available observations; for a detailed introduction into this topic see Section 1 of Paper III.

Biodiversity

Biodiversity is a concept to describe the variability of biological organisms on Earth; for an introduction on the subject see Colwell (2009) and the references therein. In order to actually quantify biodiversity from observations, certain aspects of biodiversity are described with biodiversity indices, which are functions of observable qualities. The simplest biodiversity index is *richness*, which is defined as the number S of distinct species in an area. For example, richness was used by Gaston (2000) to describe global biodiversity patterns as a function of latitude, elevation, and annual precipitation. Another aspect of biodiversity is captured by the relative *abundances* of species $\boldsymbol{p} = (p_1, \ldots, p_S)^\top$, where $p_i \in [0, 1]$ denotes the proportion of individuals classified as species i. Several mathematical functions are used to summarize $\boldsymbol{p}$ into one number. Examples of commonly used indices are the Shannon index $H(\boldsymbol{p})$ (Shannon, 1948), the Simpson index $D(\boldsymbol{p})$ (Simpson, 1949), and the species evenness index $J(\boldsymbol{p})$ (Hill, 1973). They originate from the field of information theory and are defined as

$$H(\boldsymbol{p}) = -\sum_{i=1}^{S} p_i \ln p_i \,, \qquad D(\boldsymbol{p}) = 1 - \sum_{i=1}^{S} p_i^2 \,, \qquad J(\boldsymbol{p}) = \frac{H(\boldsymbol{p})}{H_{\max}} = \frac{H(\boldsymbol{p})}{\ln S} \,.$$

Figure 1 shows the behavior of those three indices in the case where $S = 2$ species are considered. Common to all depicted indices is that their values increase as the proportions p_1 and p_2 become more similar and have their maximum at $p_1 = 1/2 = p_2$. For other values of p_1 and p_2 their values differ because of distinct weighting schemes of richness and abundance. Recent efforts tried to ease the interpretation of biodiversity indices by describing them in a unified framework including $H(\boldsymbol{p})$, $D(\boldsymbol{p})$, and $H(\boldsymbol{p})$ as special cases (Jost, 2006; Tuomisto, 2010).

The spatial scale at which biodiversity indices are calculated is closely linked to their values. One reason for this is the possibility to detect new species whenever the study area is increased leading to a larger richness S for larger areas. It is therefore crucial to calculate biodiversity indices at a common scale if the goal is to compare different regions against each other. Alternatively, one can model the link between spatial scales and richness with a species–area relationship (Tjørve and Tjørve, 2001). Also the concept of α, β, and γ diversity is related to spatial scales and provides a mean to characterize landscape diversity from diversity measurements within and among certain habitats (Whittaker, 1972). Moreover, the underlying classification of organisms into classes is essential. That classification is often done via phylogenetic and functional criteria and should be identical whenever values from different areas are compared (Cadotte et al., 2011).

Abundance based species indices are only one family of the *essential biodiversity variables*, which are proposed to assess biodiversity (Pereira et al., 2013; Proença et al., 2016). Another important family of variables is based on remotely sensed observations and has the advantage of covering an extensive spatial and temporal extent of the Earth. In particular, remotely sensed measurements can be used to describe habitat changes (Turner, 2011) and to assess species richness through productivity–biodiversity patterns (Virtanen et al., 2013). Moreover, the remotely sensed spatial heterogeneity patters can be linked to species diversity estimates from field observations (Rocchini et al., 2010; Jones et al., 2014; Madritch et al., 2014).

1.2 Climate and vegetation changes in the Arctic

Average temperature in the Arctic has increased by 2°C during the time period from 1979–2010, which is above global average (IPCC, 2013, see graphs for the HadCRUT4 climate product, p. 880). Other aspects of the Arctic climate change are cloud cover (Chapin et al., 2005), precipitation and snow coverage (Serreze et al., 2000), as well as sea ice coverage (Barnhart et al., 2016); see Hinzman et al. (2013) and ACIA (2005) for a more complete overview. Although the mentioned phenomena can be measured individually, it is necessary to understand the interactions among them in order to predict the future development of specific components (Hinzman et al., 2005). Figure 2 illustrates some important processes of the Arctic ecosystem. Many of them are not only affected by but also feedback to global climate. For example, increased temperatures cause permafrost thawing and trigger the release of carbon stored in the Arctic soils. The released carbon enters the atmosphere as the greenhouse gas CO_2 and, in turn, contributes to increased temperatures (Oechel et al., 1993; Zimov et al., 2006; Limpens et al., 2008).

In order to improve the understanding of changes in the Arctic ecosystem—and by implication its impacts on global climate—many studies have investigated ecosystem processes. In the following, we mention insights in important processes involving vegetation. Note that it is difficult to structure the scientific contributions because many of them are concerned with several processes and combine observations from different sources. Nevertheless, we tried to group them according to the predominant type of observations that contributed to the findings:

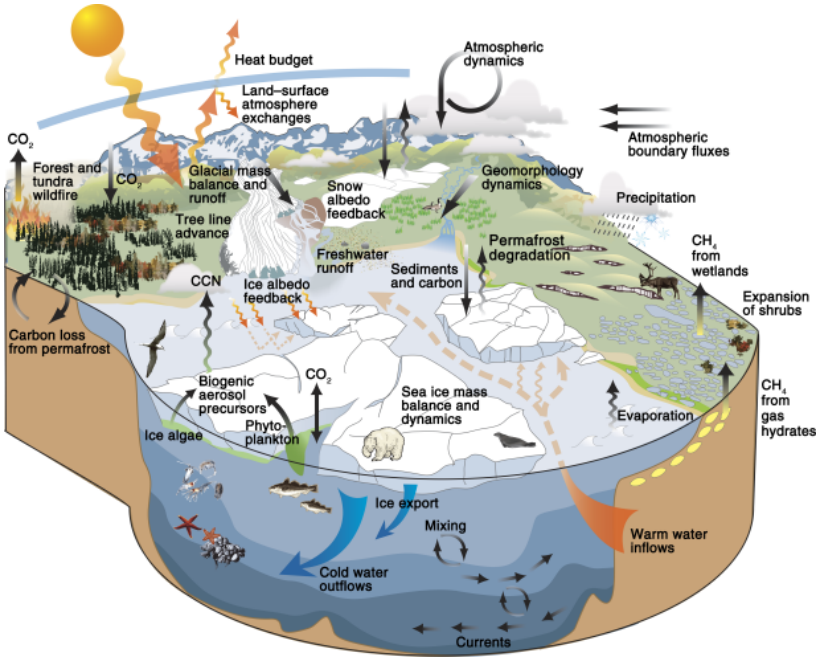


Figure 2: *Interactions among important Arctic ecosystem components and their feedback paths to climate. Source: Hinzman et al. (2013).*

Field observations and intervention experiments are used to measure shrub expansion in the Arctic tundra (Blok et al., 2010; Myers-Smith et al., 2011, 2015). Shrubification is a consequence of increased summer temperatures, which affects snow-free periods and interacts with the surface albedo, as well as with energy and water balances (Blok et al., 2011; Chapin et al., 2005; Sturm et al., 2005; Juszak et al., 2016). Similar latitudinal and elevation shifts of species were also observed for non-Arctic regions (Chen et al., 2011a). Changes in shrub cover imply changes in species composition and biodiversity, which were investigated with warming experiments (Lang et al., 2012; Elmendorf et al., 2012a,b). To enable statements about larger areas often field observations from several locations and research groups are jointly analyzed. An effort to simplify such analyses is the AVA, which was launched in 2013 and aims at collecting and harmonizing filed measurements in an open-access data base (Walker et al., 2016).

Remote sensing is another mean to monitor vegetation. In accordance with the observed shrubification, satellite based vegetation indices show greening trends for the Arctic, which are partially explained by climate change (Pouliot et al., 2009; Bhatt et al., 2010; de Jong et al., 2013). A more pronounce greening trend was attributed to the Eurasian Arctic in comparison to the American Arctic (Bi et al., 2013). A detailed assessment of the distribution of vegetation is available by the circumpolar vegetation map, which is based on remotely sensed observations from AVHRR sensors and expert knowledge from field observations (Walker et al., 2005). Other studies use remotely sensed observations to extrapolate field measurements to circumpolar Arctic (Walker et al., 2003; Raynolds et al., 2008) and to quantify changes in surface albedo on a Pan-Arctic scale (Loranty et al., 2011).

Deterministic models help to test the current understanding of ecosystem processes and to retrieve predictions of future states of the system. For example, Juszak et al. (2014) used a Discrete Anisotropic Radiative Transfer (DART) model to quantify the energy uptake of different branch and leave configurations. Others used Global climate model outputs to investigate feedbacks mechanisms and to predict future vegetation composition (Swann et al., 2010; Pearson et al., 2013).

1.3 Which statistical developments are needed?

Many of the mentioned studies above rely on established statistical tools to analyze data. They use statistical methods range from simple t-tests to more sophisticated methods such as generalized linear mixed models (Bolker et al., 2009), structural equation models (Grace et al., 2010), and Bayesian hierarchical models (BHMs, e. g., Korner-Nievergelt et al., 2015). Aside from the fact that it can be challenging to applied these methods meaningfully, they are well understood for a wide range of situations. However, the following two features of recent ecological data generate new research questions for statistical and computational sciences, and motivate the methodological developments presented in this PhD thesis:

First, the typical dataset size is getting huge particularly when satellite observations and/or climate predictions are involved. For example, a typical Landsat 7 satellite image consists of more than 34 million pixels (30 m resolution for an approximate scene size of $170\,\text{km} \times 183\,\text{km}$; source https://landsat.usgs.gov), which is several orders of magnitude beyond the processing capabilities of classical statistical models for spatial data (e. g., classical kriging outlined in Cressie 1990). In order to work towards statistical models for such datasets, the following two approaches are promising: On the one hand, traditional spatial models and their software implementation can be developed further to make them capable of processing larger amounts of data. The research area concerned with that strategy is detailed in Subsection 2.1 and novel contributions to it are presented in Paper I and Paper II. On the other hand, one can focus on the parallel computing mode of large computer systems and develop spatio-temporal models that are explicitly designed to exploit such resources. A new prediction method implementing this strategy is presented in Paper III.

Second, state-of-the-art ecological data feature the two fundamentally different data types introduced above, namely field measurements that are precise but sparse in space and time, and remotely sensed observations that have an extensive spatio-temporal coverage but low precision. Summarizing those data types into a combined data product can lead to more precise information with extensive coverage. One way to achieve this is data fusion based on multi-resolution BHMs; see also Subsection 2.2 for more information. As computational limitations constitute a major drawback for such models, we investigate different inference strategies in Paper VI.

2 Space-time statistics

Space-time statistics is used to process and analyze data describing the evolution of a process in space and time (Mateu et al., 2003). Besides the measurements of interest, such data feature information about where and when the measurements were taken. Example data types are field measurements of vegetation and meteorological conditions, remotely sensed satellite observations, and climate model outputs. The reason why specialized statistical techniques are required to model such data is their typical correlation structure, which is nicely described by Tobler (1970): "everything is related to everything else, but near things are more related than distant things". To account for and exploit this correlation, spatio-temporal models are required.

Recorded information about the spatial and temporal position of measurements are always of discrete nature. However, there are two fundamentally different ways to think about them. On the one hand, they can be seen as a realization of the spatio-temporal process $Y(\cdot)$ defined as

$$\{Y(s) : s \in \mathcal{D} \subset \mathbb{R}^d,\ d \in \mathbb{N}\}.$$

In that framework $Y(\cdot)$ is modeled at every spatio-temporal location s in the not necessarily discrete domain $\mathcal{D}$. Note that this is not in contradiction to the fact that the models are always informed by a finite amount of observations. Statistical models for this data type are called process models and are typically used for data recorded at irregularly spaced location in time and space; such as vegetation measurements taken in the field. On the other hand, spatio-temporal data can be modeled in a discrete space framework, in which data are described at a finite amount of locations only. Such models are termed lattice models and describe the data with the multivariate random vector

$$\mathbf{Y} = (Y(s_1), \ldots, Y(s_n))^\top \in \mathbb{R}^n,\ s_i \in \mathbb{R}^d,\ n, d \in \mathbb{N}.$$

Lattice models are usually used to model data observed on a regular grid in space and time including areal data; examples are satellite datasets and population data recorded per administrative spatial unit, respectively.

For those two views on spatio-temporal data there exists various statistical frameworks to infer model parameters from data. We distinguish between frequentist, Bayesian, and distribution-free models (Stuart et al., 2008). In broad terms, frequentist models make statements about possible or imaginary repetitions of the data collection experiment. Bayesian approaches use data to transform prior probabilities into posterior probabilities of the quantities of interest. Eventually, distribution-free models apply algorithmic procedures to data in order to obtain quantities with favorable properties. As spatio-temporal data are often very large, an important requirement for all inference methods is that they are computational efficient.

In the reminder of this section, we outline statistical and computational concepts that are helpful to understand the scientific contributions of this PhD thesis. Subsection 2.1 introduces a frequentist approach to spatio-temporal process models, which is the basis for Paper I and motivated the software development presented in Paper II. Subsection 2.2 is devoted to Bayesian hierarchical models and emphasizes their capability to jointly model datasets from different sources. Technical aspects of the inference for Bayesian models have motivated the work presented in Paper VI. Last but not least, Subsection 2.3 points to algorithmic procedures for predicting missing values in satellite datasets and sets the scene for Paper III.

2.1 A frequentist approach to space-time process models

One possibility to characterize process models is to use the framework of Gaussian process models, which assumes that every finite realization of the process can be described with a multivariate Gaussian distribution (Cressie and Wikle, 2015). More formally, if $Y(\cdot)$ is a zero mean Gaussian process, then any realization $\boldsymbol{y} \in \mathbb{R}^n$ thereof at a finite amount of locations $\mathbf{Y} = (Y(s_1), \ldots, Y(s_n))^\top \in \mathbb{R}^n$, $s_i \in \mathcal{D}$ has the density

$$f_{\mathbf{Y}}(\boldsymbol{y}) = \left(\frac{1}{\sqrt{2\pi}}\right)^n \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{y}\right), \tag{1}$$

where $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix and $\det(\boldsymbol{\Sigma})$ is the determinant of $\boldsymbol{\Sigma}$. Moreover, a Gaussian process $Y(\cdot)$ is second-order stationary, if the covariance of the process at any two locations in $\mathcal{D}$ can be expressed as

$$\mathrm{Cov}(Y(s_1), Y(s_2)) = c(s_1 - s_2),\ s_1, s_2 \in \mathcal{D}, \tag{2}$$

for some positive (semi-)definite covariance function $c(\cdot)$. Covariance functions can be used to derive the covariance matrix $\boldsymbol{\Sigma}$ corresponding to an arbitrary set of locations in $\mathcal{D}$. In the case where $c(\cdot)$ can be expressed as $c(|s_1 - s_2|) = c(h)$, the process is called isotropic, otherwise it is called anisotropic.

Often $c(\cdot)$ is defined as parametric functions depending on the parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^\top$. To estimate $\boldsymbol{\theta}$ from a realization of the process $\boldsymbol{y}$, we can write $\boldsymbol{\Sigma}$ as a function of $\boldsymbol{\theta}$ in the probability density function of Equation (1), which can then be interpreted as the likelihood function $f_{\mathbf{Y}}(\boldsymbol{y}, \boldsymbol{\theta})$. Hence, the maximum likelihood estimates of $\boldsymbol{\theta}$ can be obtained as $\widehat{\boldsymbol{\theta}}_{\mathrm{ML}} = \max_{\boldsymbol{\theta}} f_{\mathbf{Y}}(\boldsymbol{y}, \boldsymbol{\theta})$. Maximizing $f_{\mathbf{Y}}(\boldsymbol{y}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ requires to evaluate the density in Equation (1) many times and is computationally demanding. While for large datasets the explicit construction of $\boldsymbol{\Sigma}$ already requires considerable computing resources, the limiting bottlenecks are the evaluations of $\mathbf{y}^\top \boldsymbol{\Sigma}^{-1} \mathbf{y}$ and $\det(\boldsymbol{\Sigma})$, which are usually computed with a Cholesky factorization of $\boldsymbol{\Sigma}$. The latter constrain the dataset size that can be handled with a brute force implementation of Gaussian process models to the order of 10^3 on typical computing machines.

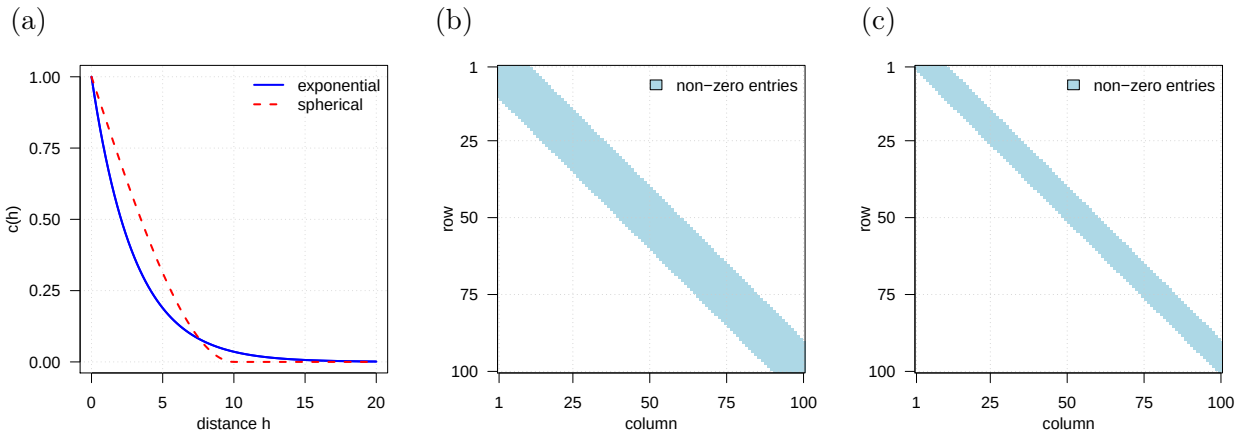


Figure 3: *(a) the exponential and spherical covariance functions specified in Equations* (3) *and* (4)*, respectively. (b) non-zero entries of the covariance matrix* $\boldsymbol{\Sigma}_{\text{sph}}$*. (c) non-zero entries of the upper Cholesky factor of* $\boldsymbol{\Sigma}_{\text{sph}}$*.*

Because of this computational limitation, specialized model designs have been developed, which allow applying Gaussian process models to datasets of the orders 10^5 to 10^6. One strategy to make computations more efficient relies on Kronecker formulations and separable models (Gentle, 2007; Furrer and Genton, 2011). Other use different types of approximation such as low-rank models (Cressie and Johannesson, 2008; Stein, 2008), composite likelihood approaches (spatial: Vecchia, 1988; Stein et al., 2004; Eidsvik et al., 2014; space-time: Varin et al., 2011; Bevilacqua et al., 2012), and Gaussian Markov random field (GMRF) type approximations (Rue and Held, 2005; Hartman and Hössjer, 2008; Lindgren et al., 2011). Other approximation techniques include predictive process (Banerjee et al., 2008; Finley et al., 2009), discrete process convolution (Higdon, 2002; Lemos and Sansó, 2009), lattice kriging (Nychka et al., 2015), modeling in the spectral domain (Fuentes, 2007), and covariance tapering (Furrer et al., 2006). For reviews of geostatistical approaches for large datasets see Sun et al. (2012) and Bradley et al. (2016).

Many of these efficient models rely on sparse covariance matrices $\boldsymbol{\Sigma}$, where sparse indicates that the matrix contains many zero elements. Working with sparse matrices reduces the computational workload to evaluate the density given in Equation (1) via specialized storage formats (Eisenstat et al., 1982; Golub and Loan, 1996) and adapted algorithms for the Cholesky factorization (Ng and Peyton, 1993; Gilbert et al., 1994; Chen et al., 2008). To run such algorithms fast they are typically implemented in low-level programming languages such as C/C++ and Fortran. As low-level languages are inconvenient when it comes to manipulating data, such programs often come along with a user interface written in a high-level programming language like the open source software R (2017).

We illustrate the computational gains realized when using sparse covariance matrices in Gaussian process models with the following example. Let $Y(\cdot)$ be a second-order stationary and isotropic Gaussian process defined on $\mathcal{D} = [0, 100] \subset \mathbb{R}$. Then the correlation structure of $Y(\cdot)$ can be defined with the exponential type covariance function as, e. g.,

$$c_{\text{exp}}(h) = \exp\left(-\frac{|h|}{3}\right),\ h = |s_i - s_j|\,; \tag{3}$$

see also Figure 3-a. If we consider the 100 equally spaced locations $\boldsymbol{s} = (s_1, \ldots, s_{100})^\top = (1, \ldots, 100)$ in $\mathcal{D}$, we can derive the corresponding covariance matrix $\boldsymbol{\Sigma}_{\text{exp}}$ describing the correlation of $\mathbf{Y} = (Y(s_1), \ldots, Y(s_{100}))^\top$. As $c_{\text{exp}}(h) > 0, \forall h \in \mathbb{R}$, it follows that the corresponding 100×100 covariance matrix $\boldsymbol{\Sigma}_{\text{exp}}$ as well as the Cholesky factor thereof have $10'000$ non-zero entries.

Another possible choice for the covariance function is a spherical type covariance function, e. g., defined as

$$c_{\text{sph}}(h) = \begin{cases} 1 - \frac{2}{3}\frac{|h|}{10} + \frac{1}{2}\frac{|h|}{10}, & |h| \leq 10, \\ 0, & |h| > 10, \end{cases} \tag{4}$$

and shown in Figure 3-a. According to that definition, each $Y(s_i)$ is uncorrelated to $Y(s_j)$ if and only if the distance between s_i and s_j is greater than 10. As a consequence, the corresponding covariance matrix $\boldsymbol{\Sigma}_{\text{sph}}$ describing the correlation structure of $\mathbf{Y}$ contains only $1'810$ $(18,1\%)$ non-zero entries, which is significantly less compared to the $10'000$ non-zero entries of the previously considered $\boldsymbol{\Sigma}_{\text{exp}}$. The matrix $\boldsymbol{\Sigma}_{\text{sph}}$ has the banded structure shown in Figure 3-b. Moreover, the Cholesky factor of that $\boldsymbol{\Sigma}_{\text{sph}}$ is usually sparse depending on the chosen pivot permutation; see Figure 3-c for one possible sparsity structure of the Cholesky factor of $\boldsymbol{\Sigma}_{\text{sph}}$.

In order to exploit the sparsity structure of $\boldsymbol{\Sigma}_{\text{sph}}$ computationally, specialized algorithms are used, which store only the non-zero entries of the matrices. One software option providing such a storage format is the R package *spam* (Furrer and Sain, 2010). It allows us to construct $\boldsymbol{\Sigma}_{\text{sph}}$ with the following R code:

```
R> library("spam")
R> S <- 1:100
R> h <- nearest.dist(S, delta = 1000, upper = NULL)
R> Sigma.sph <- as.spam(cov.sph(h, c(10, 1, 0)))
```

A closer look at the structure of `Sigma.sph` reveals insights into the storage format.

```
R> str(Sigma.sph)
Formal class 'spam' [package "spam"] with 4 slots
  ..@ entries    : num [1:1810] 1 0.85 0.704 0.564 0.432 ...
  ..@ colindices : int [1:1810] 1 2 3 4 5 6 7 8 9 10 ...
  ..@ rowpointers: int [1:101] 1 11 22 34 47 61 76 92 109 127 ...
  ..@ dimension  : int [1:2] 100 100
```

In words, the slot `entries` contains the values of the non-zero elements of `Sigma.sph` and `colindices` and `rowpointers` encodes their positions in the matrix. This storage format and the specialized algorithm are most efficient if matrices contain many zero elements. For example, $\mathbf{\Sigma}_{\text{sph}}$ with its 18.1% non-zero entries is not sparse enough to compensate the computational overhead of the Cholesky factorization for sparse matrices. In fact, using the ordinary matrix format of R together with the corresponding Cholesky factorization is about tree times faster compared to the sparse version. However, if we increase the number of considered locations from 100 to 2′000, the resulting $\mathbf{\Sigma}_{\text{sph}}$ has only 1% non-zero entries and the *spam* version outperforms the ordinary Cholesky algorithm by a factor of 270.

In this PhD thesis we present two novel contributions to the field of likelihood based Gaussian process models. The first contribution is that we relaxed the second-order stationary assumption (Equation (2)) of Gaussian process models by introducing a covariance function that depends on observed spatial covariates. Using that covariance function we were able to obtain a spatially varying description of the covariance of satellite based NDVI measurements. Please see Paper I for more information. The second contribution concerns the software implementation of sparse matrices in the R package *spam*. More precisely, we extended *spam* to exploit the 64-bit capabilities of R by enabling sparse matrices with more then $2^{31} - 1$ non-zero entries. As R does not expose a proper 64-bit data type to its R application programming interface, wrapper functions were required. This motivated the development of the R package *dotCall64*. Technical insights into that package are presented in Paper II.

2.2 Bayesian hierarchical models for lattice data

Models for lattice data differ from the previously considered process models in that they only model a finite number of spatio-temporal locations. This simplifies the conceptual framework as one can work with random vectors of known length from the beginning. Another change in the setting is that we now rely on Bayesian inference instead of using frequentist likelihood models. In the following we present relevant background information, which contributes to a better understanding of Paper VI. Thereby we emphasize that Bayesian hierarchical models (BHMs) are a promising tool to fuse information from Arctic field measurements with remotely sensed observations.

In a Bayesian setting data modeling follows three main steps: First, a likelihood (model) is developed, which describes the distribution of the data based on the parameters $\boldsymbol{\theta}$. Second, the prior believes (current knowledge) of each parameter of the likelihood is summarized with a distribution. Third, the inference step combines the data and the prior distributions. As a result, the updated knowledge of the likelihood parameters is reflected in posterior distributions. Based on classical probability calculus, one can show the following proportional relation between the posterior, likelihood, and prior distributions

$$\underbrace{\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}_{\text{posterior}} \propto \underbrace{\pi(\boldsymbol{y} \mid \boldsymbol{\theta})}_{\text{likelihood}} \underbrace{\pi(\boldsymbol{\theta})}_{\text{prior}}, \tag{5}$$

where $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ denotes a vector of observed data values, $\pi(\cdot)$ a density, $\pi(\cdot \mid \cdot)$ the conditional density of the first argument given the second, and $\propto$ the proportional sign. In the next paragraphs, we discuss important aspects of BHMs by means of an example and refer to one of the many books about Bayesian statistics for a more detailed introduction (e. g., Bolstad, 2007; Albert, 2007).

Let $\boldsymbol{y} \in \mathbb{R}^n$ be a vector of n observed data values, which are assumed to be well described by the multivariate normal distribution $\mathcal{N}\big(\mu\mathbf{1},\, \tau^{-1/2}\mathbf{I}\big)$, where $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$ is a vector of ones, $\mathbf{I} = \mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{n \times n}$ is the diagonal matrix with ones on the diagonal, and $\mu, \tau \in \mathbb{R}$, $\tau > 0$ are the model parameters to be inferred from $\boldsymbol{y}$. To complete the model let μ follow the Gaussian distribution $\mathcal{N}(\alpha_\mu, \beta_\mu)$ and τ the gamma distribution $\Gamma(\alpha_\tau, \beta_\tau)$, where $\alpha_\mu, \beta_\mu, \alpha_\tau, \beta_\tau \in \mathbb{R}$ are fixed (non-random) hyper-priors and chosen such that the distributions are well-defined. The described model is summarized by the directed acyclic graph $\mathcal{G}$ shown in Table 1. The graph $\mathcal{G}$ encodes the dependency structure of the variables; e. g., it shows that $\boldsymbol{y}$ depends only on μ and τ. According to that dependency structure, the joint probability distribution can be factorized as

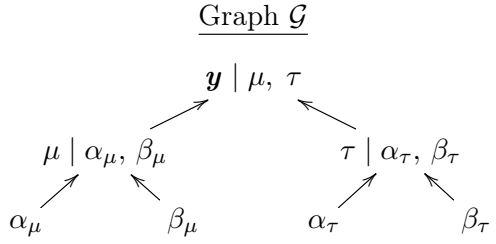
$$\pi(\boldsymbol{y}, \mu, \tau, \alpha_\mu, \beta_\mu, \alpha_\tau, \beta_\tau) \propto \pi(\boldsymbol{y} \mid \mu, \tau)\, \pi(\mu \mid \alpha_\mu, \beta_\mu)\, \pi(\tau \mid \alpha_\tau, \beta_\tau). \tag{6}$$

Suppose we aim at estimating the posterior distributions of μ given the hyper-priors and the data. If the latter are considered as fixed components of the joint distribution given in Equation (6), the remaining random parameters are μ and τ. Hence, determining the marginal posterior distribution of μ leads to the following integration problem

$$\pi(\mu \mid \boldsymbol{y}, \alpha_\mu, \beta_\mu, \alpha_\tau, \beta_\tau) = \int \pi(\mu, \tau \mid \boldsymbol{y}, \alpha_\mu, \beta_\mu, \alpha_\tau, \beta_\tau)\, d\tau.$$

That integral and many other integrals emerging in Bayesian inference have no closed form solution. Thus, it is necessary to rely on computationally expensive methods such as Markov chain Monte Carlo sampling (see Robert and Casella, 2004, for an overview) and approximation techniques (e. g., integrated nested Laplace approximations, Rue et al. 2009). Specialized software packages facilitate the implementation of sampling methods for common model structures (Carpenter et al., 2017; Lunn et al., 2012; Plummer, 2003). However, more complex models are often not supported

Table 1: *Summary of the discussed Bayesian hierarchical model.*



Levels	Distributions
likelihood	$\mathcal{N}\big(\mu\mathbf{1},\, \tau^{-1/2}\,\mathbf{I}\big)$
priors	$\mathcal{N}(\alpha_\mu,\, \beta_\mu)$, $\Gamma(\alpha_\tau,\, \beta_\tau)$
hyper-prior	fixed

by those programs or their sampling methods are too slow for the large dataset sizes. As a consequence, it can be necessary to implement samplers from scratch or to extend existing software packages. Paper VI provides annotated R code for different implementation strategies of BHMs and compared them against each other. Thereby, surprisingly large differences between different types of implementations were detected.

In the example above the observations in $\boldsymbol{y}$ are independent of each other. However, many data exhibit spatio-temporal correlation structures, which need to be taken into account to obtain accurate models. Often such dependencies are modeled with the likelihood or an additional introduced prior level (also called process level). Following the former strategy, we could replace the likelihood with $\mathcal{N}(\mu\mathbf{1},\, \boldsymbol{\Sigma}(\boldsymbol{\lambda}))$, where $\boldsymbol{\Sigma}(\boldsymbol{\lambda})$ is a symmetric positive definite $n \times n$ matrix. The parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)^\top$ can be used to control certain aspects of the correlation structure and can be inferred from the data when appropriate prior distributions are specified. Often it is computationally beneficial to work with Gaussian Markov random fields (GRMFs) and parameterize the precision matrix $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$ instead of $\boldsymbol{\Sigma}^{-1}$ (Rue and Held, 2005; Lindgren et al., 2011). Also the models discussed in Paper VI use GMRFs to model spatial dependencies. However, in contrast to the data considered in the example, the paper discusses the case where the data are count data and the likelihood follows a Poisson distribution instead of Gaussian one. This makes the derivations and computations more challenging.

BHMs for data fusion

Bayesian hierarchical models can be used to jointly model data that are recorded at several spatio-temporal resolutions and exhibit multiple measurement error structures. The described task is known as "data fusion" and sometimes termed "down scaling" or "up scaling" depending on whether the resolution of the field of interest is larger or smaller compared to the resolution of the input data. Data fusion with BHMs was investigated in the course of this PhD, as it is a promising tool to combine the available information about the Arctic vegetation (Schaepman-Strub et al., 2013). In the following, we illustrate the method with a simulated data example and discussion extensions of it with references to published models.

The simulation example is concerned with estimating the NDVI values $z_1, \ldots, z_m$ at m regularly spaced locations along a transect $S \subset \mathbb{R}$. The field $\boldsymbol{z} \in \mathbb{R}^m$ is not observed directly but through two types of measurements: One type consists of ground data $\boldsymbol{y}_g \in \mathbb{R}^{n_g}$, which are taken at n_g distinct locations in S. They have a small spatial support resulting in (almost) punctual measurements of $\boldsymbol{z}$. The values $\boldsymbol{y}_g$ do not exactly coincide with the corresponding values of $\boldsymbol{z}$ because of measurement errors. The other measurement type is remote sensing based and provides gridded observation for the entire transect S at coarse resolution. We denote those measurements with $\boldsymbol{y}_r \in \mathbb{R}^{n_r}$ and assume that they consist of locally averaged values of $\boldsymbol{z}$ (one for each grid cell) and measurement error. The left panel of Figure 4 illustrates $\boldsymbol{z}$, $\boldsymbol{y}_g$, and $\boldsymbol{y}_r$.

The task of the BHM is to infer $\boldsymbol{z}$ from $\boldsymbol{y} = (\boldsymbol{y}_g^\top, \boldsymbol{y}_r^\top)^\top \in \mathbb{R}^n$, where $n = n_g + n_r$. To that end,

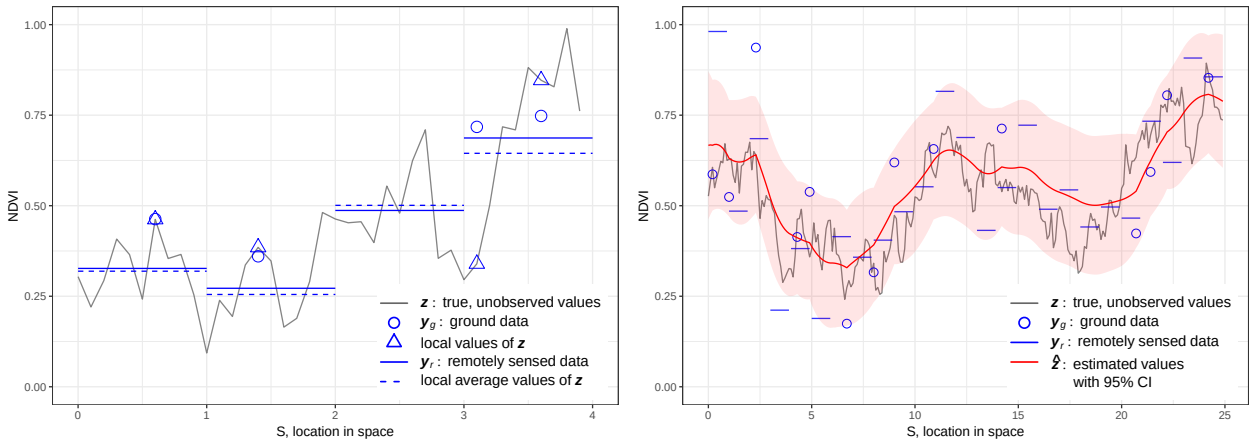


Figure 4: *Simulated data example showing the fusion of two data types on a transect S (x-axes). The goal is to estimate the NDVI values $\boldsymbol{z}$ (gray line) from ground data $\boldsymbol{y}_g$ (blue circles) and gridded remotely sensed data $\boldsymbol{y}_r$ (blue lines). Left: data generation and measurement errors of $\boldsymbol{y}_g$ and $\boldsymbol{y}_r$. Right: BHM based estimates of the NDVI field $\widehat{\boldsymbol{z}}$ (red line) together with its 95% credibility interval (red area).*

we describe the link between $\boldsymbol{y}$ and $\boldsymbol{z}$ in the likelihood level of the BHM and define the function $\mathcal{H} : \mathbb{R}^m \to \mathbb{R}^n$, which maps $\boldsymbol{z}$ to the observed values $\boldsymbol{y}$. More precisely, single values of $\boldsymbol{z}$ are mapped to the corresponding values of $\boldsymbol{y}_g$ with an identity function. Conversely, mean values of $\boldsymbol{z}$ taken over the corresponding grid cells of the remotely sensed data are mapped to the corresponding values of $\boldsymbol{y}_g$. Both relations can be expressed as left multiplication of $\boldsymbol{z}$ with the $n \times m$ matrix $\mathbf{H}$. Moreover, the measurement errors of $\boldsymbol{y}_g$ and $\boldsymbol{y}_r$ are modeled with Gaussian distributions having the variances σ_g^2 and σ_r^2, respectively. In summary, the likelihood is

$$\boldsymbol{y} \mid \boldsymbol{z},\ \sigma_g,\ \sigma_r \sim \mathcal{N}(\mathbf{H}\boldsymbol{z},\ \mathbf{D}(\sigma_g,\ \sigma_r)), \quad \mathbf{D}(\sigma_g,\ \sigma_r) = \operatorname{diag}(\underbrace{\sigma_g^2, \ldots, \sigma_g^2}_{n_g}, \underbrace{\sigma_r^2, \ldots, \sigma_r^2}_{n_r}), \tag{7}$$

where $\operatorname{diag}(\,\cdot\,)$ denotes a diagonal matrix with the elements inside the parenthesis on the diagonal.

The values of $\boldsymbol{z}$ (and hence $\boldsymbol{y}$) features a spatial dependency structure, which is taken into account by letting $\boldsymbol{z}$ follow a multivariate Gaussian distribution with mean zero and covariance $\boldsymbol{\Sigma}(\boldsymbol{\lambda})$. (As NDVI values per definition have a mean greater than zero, we mean-center the data before fitting the model and back-transform the results afterwards. Including a mean term into the model would be another option.) The covariance $\boldsymbol{\Sigma}(\boldsymbol{\lambda})$ is a symmetric positive definite $m \times m$ matrix and can capture different spatial dependency structures depending on the values of $\boldsymbol{\lambda}$. All the parameters $\boldsymbol{\lambda}$, σ_g, and σ_r can be informed by the data when suitable prior distributions are specified. For simplicity, however, we keep them at fixed values in this illustration. Table 2 shows a summary of the described BHM.

According to the graph $\mathcal{G}$ in Table 2 the joint probability distribution of all parameters is

$$\pi(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\lambda}, \sigma_g, \sigma_T) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{y} - \mathbf{H}\boldsymbol{z})^\top \mathbf{D}(\sigma_g,\ \sigma_r)^{-1}(\boldsymbol{y} - \mathbf{H}\boldsymbol{z}) - \frac{1}{2}\boldsymbol{z}^\top \Sigma(\boldsymbol{\lambda})^{-1}\boldsymbol{z}\right\}. \tag{8}$$

Hence, the posterior distribution of $\boldsymbol{z}$ given the observed data $\boldsymbol{y}$ and the fixed priors follows the Gausian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{S})$ with

$$\boldsymbol{\mu} = \boldsymbol{y}^\top \mathbf{D}(\sigma_g, \sigma_r)^{-1}\mathbf{H}\mathbf{S}, \qquad \mathbf{S} = \left(\mathbf{H}^\top \mathbf{D}(\sigma_g, \sigma_r)^{-1}\mathbf{H} + \Sigma(\boldsymbol{\lambda})^{-1}\right)^{-1}. \tag{9}$$

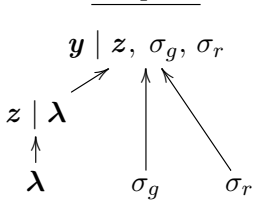
Note that in this setting the posterior distribution is available in closed form and no computationally demanding sampling techniques are needed to characterize it.

The right panel of Figure 4 illustrates the model with a simulated data example, which was generated as follows: First, a realization of the true and unobserved process $\boldsymbol{z}$ (black line) was simulated using a Gaussian distribution with an exponential covariance model. In the next step, both the gridded observations (blue lines) and point measurements (blue dots) were generated based on $\boldsymbol{z}$ and under the assumption of normally distributed measurement errors. Eventually, the introduced model was fitted to the data to get an estimate of $\boldsymbol{z}$ (red line) together with a 95% credibility interval (red area). As expected, the credibility interval covers most of the values of $\boldsymbol{z}$.

In a more realistic data example the following aspects and extensions can be considered:

- The parameters σ_g, σ_r, and $\boldsymbol{\lambda}$ are often not known and have to be estimated from the data. In that case the model can be extended by replacing the fixed values of those parameters with suitable prior distributions. As a consequence, the posterior distribution has no longer a closed form expression and computationally expensive inference methods are required. Often MCMC method are used to sample from the posterior distribution (e. g., Wilson et al., 2011). Alternatively, the expectation maximization algorithm can be used (Bolin et al., 2009; Finazzi et al., 2013).
- In the presented model the measurement errors of $\boldsymbol{y}$ are normally distributed, which leads to a convenient form of the posterior distribution. However, depending on the structure of the observed data, it can be beneficial to chose other distributions. For example, count data can be modeled with a Poisson and proportions with a binomial distribution. Such setting require more elaborate sampling procedure; see Schliep and Hoeting (2013) and Paper VI (p. 97) for examples.

Table 2: *Summary of the discussed Bayesian hierarchical data fusion model.*



Graph $\mathcal{G}$	Levels	Distributions
$\boldsymbol{y} \mid \boldsymbol{z},\ \sigma_g,\ \sigma_r$	likelihood	$\mathcal{N}(\mathbf{H}\boldsymbol{z},\ \mathbf{D}(\sigma_g,\ \sigma_r))$
$\boldsymbol{z} \mid \boldsymbol{\lambda}$	process	$\mathcal{N}(\mathbf{0},\ \boldsymbol{\Sigma}(\boldsymbol{\lambda}))$
$\boldsymbol{\lambda}$ σ_g σ_r	prior	fixed

- The mapping $\mathcal{H}(\cdot)$ as defined above is a linear transformation. The generalization to non-linear transformation complicates the derivation of the posterior distribution and inference (Gryparis et al., 2007; Bliznyuk et al., 2014).
- The example above links ground measurements and remotely sensed data. However, $\mathcal{H}(\cdot)$ can be used to link more and different data types to $\boldsymbol{z}$. For example, it is possible to link two satellite products (Chakraborty et al., 2015) and to incorporate model outputs from physical simulations (Berrocal et al., 2012).
- The correlation structure of $\boldsymbol{z}$ can be extended to capture more complex dependencies. Most studies consider two spatial and one temporal dimension (e. g., Finazzi and Fassò, 2014; Sahu et al., 2007). Moreover, it is possible to go beyond modeling correlations by including knowledge about the physical behavior of the process of interest (Sigrist et al., 2012).
- Finally, computationally intensive inference techniques limit the dataset sizes that can be handled. A promising approach to reduce the computational workload is to introduce an auxiliary lattice (Sahu and Bakar, 2012; Xu et al., 2015; Nychka et al., 2015).

2.3 Distribution-free methods for satellite data

All statistical models discussed so far make assumptions about the distributions of the considered data. Relaxing these assumptions leads to distribution-free models, which are also termed non-parametric models in some contexts (Stuart et al., 2008). A prominent example of a distribution-free model in spatial statistics are smoothing splines (Wahba, 1990). Other methods were specifically designed for the prediction of missing values in spatio-temporal satellite datasets. An example is the "neighborhood similar pixel interpolate" method, which uses weighted values from images observed at other points in time (Chen et al., 2011b; Weiss et al., 2014); see also the introduction of Paper III (p. 79) for a more complete overview.

Low quality or missing values occur in many datasets from optical satellite products for reasons like cloud cover. In the field of vegetation mapping missing values constitute a major drawback (Stow et al., 2004; Hmimina et al., 2013). To illustrate the issue, we consider the MOD13A1 data product, which is based on observations collected by the Moderate Resolution Imaging Spectroradiometer (MODIS, Didan et al., 2015). The data product has global coverage at a spatial resolution of 500 m and provides one image every 16 days. Low quality values can be identified by the included quality data layer (Roy et al., 2002). The left panel of Figure 5 shows a small subset of an NDVI dataset with low quality values depicted as black pixels. To enable data analysis methods that require complete datasets, the missing values are predicted from the observed ones by gap-filling algorithms.

In the following we give insights into the R package *gapfill* (Gerber, 2017), which can be used to predict missing values in satellite data. The package implements a modular framework to apply a wide range of distribution-free method and can be seen as a generalization of many methods similar to the "neighborhood similar pixel interpolate" method. Likewise, the gap-filling method presented in Paper III is a specific configuration of the strategy implemented in *gapfill*.

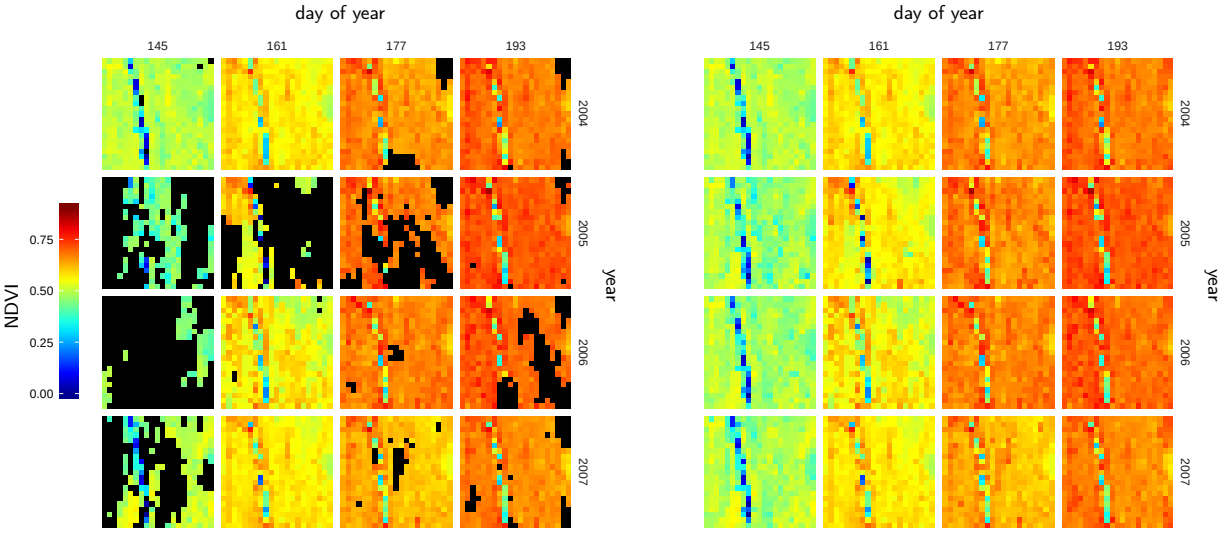


Figure 5: *Left: subset of a MODIS NDVI data product. Low quality values are shown as black pixels. Right: gap-filled version of the same dataset using a simplistic prediction algorithm.*

Define a prediction algorithm using R package *Gapfill*

The main function of the package is `Gapfill()` and its task is to predict the missing values in the input dataset. The input dataset has to be an array with four dimensions, where the dimensions typically correspond to latitude, longitude, day of the year, and year. The missing values are predicted independent of each other. This enables parallel execution of the prediction, which is implemented using the R package *foreach* (Analytics and Weston, 2014) and can be linked with OpenMP (2016) and MPI (2016) environments.

To predict one missing value, `Gapfill()` relies on a `Subset()` and a `Predict()` function, which can be specified by the user. First, the `Subset()` function selects a spatio-temporal subset of the data. Then the `Predict()` function predicts the missing value based on that subset or returns a missing value. In the latter case, the algorithm jumps back to the selection step and `Subset()` returns another subset, which is again used by `Predict()` for a second try. That cycle is repeated until `Predict()` returns the predicted value. The advantage of this iterative procedure is that the subsets selected for prediction are adaptive to local characteristics of the data, such as, e. g., the distribution of missing values.

We illustrate the procedure with an R example. First, we load the package and plot the NDVI dataset shown Figure 5 (left panel). The data are stored in the `ndvi` object available through *gapfill*.

```
R> library("gapfill")
R> Image(ndvi)
```

Next, we define a `Subset()` function, which selects different subsets from the data depending on the argument `i`.

```
R> Subset <- function (data, mp, i)
+     ArrayAround(data = data, mp = mp, size = c(i, i, 0, Inf))
```

The function `ArrayAround()` is defined in the R package *gapfill* and extracts a subset from the data, which is centered around the missing value at position `mp`. The argument `size` sets the amount of values that are included in both directions of all four dimensions. For instance, if `i=1`, `Subset()` returns an array containing 3×3 pixel images recorded at the same day of the year as the considered missing value. For larger values of `i` the spatial extent of the images is increased.

As prediction function, we define `Predict()`, which simply returns the mean all values in the subset `a`. If all values in `a` are missing, `Predict()` returns a missing value.

```
R> Predict <- function (a, i) mean(a, na.rm = TRUE)
```

Now, we can gap-fill the NDVI dataset via

```
R> out <- Gapfill(data = ndvi, fnSubset = Subset, fnPredict = Predict)
```

The results are visualized in the right panel of Figure 5. Although the defined `Subset()` and `Predict()` functions are relatively simple, the resulting gap-filled data reconstruct the seasonal trend and local features of the dataset such as the band of small values crossing the images from the top to the bottom.

As shown in the R example above, the gap-filling method implemented in the R package *gapfill* allows users to adapt the method to specific datasets with little efforts. Moreover, it contains more elaborate instances of `Subset()` and `Predict()` functions, which lead to even more accurate predictions. The algorithm was tested using an Arctic MODIS NDVI dataset. In comparison to two state-of-the-art gap-fill methods it provided the most accurate predictions; see Paper III (p. 79) for more information.

3 Research questions and outline

Understanding the Arctic vegetation and its interactions with climate are challenging scientific problems. One key aspect is that investigating these problems typically involves the analyses of very large datasets in order to make full use of the available data. Hence, harnessing statistical methods and computational implementations is necessary. This PhD thesis addresses relevant aspects thereof and is centered around the following five research questions:

R1: How can we extend the sparse matrix algebra R package *spam* to 64-bit matrices in order to model large remote sensing based data products?

R2: What is an efficient 64-bit extension of the foreign function interface of R?

R3: How can we design a flexible software to predict missing values in large spatio-temporal datasets?

R4: What are efficient implementation and validation strategies for complex Bayesian hierarchical models for spatial data?

R5: Which correlation patters between biodiversity indices derived from Arctic field measurements and landscape characterizations based on remotely sensed observations can be established?

Each of the research questions is addressed in one of the papers presented later in this thesis. For a detailed outline of the papers and the authors contributions see pages 39–43.

Applied statistics stands on scientific applications, statistics, and computing; see Figure 6 for a schematic overview. In the context of this thesis the scientific applications are concerned with Arctic ecology and related data types as introduced in Section 1. The statistics methodology covers frequentist, Bayesian, and distribution-free models as explained in Section 2. We have chosen the computational methods to fully exploit the statistical methods.

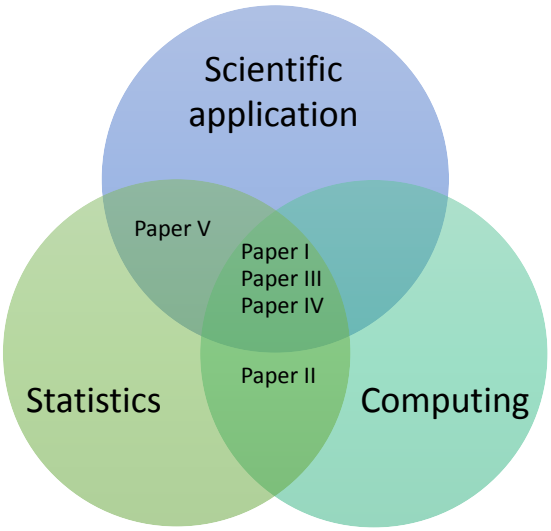


Figure 6: *Venn diagram showing the three scientific disciplines of applied statistics and the positioning of the five papers within them.*

All presented papers can be place into overlapping areas of the mentioned disciplines (Figure 6). Papers I, III and IV feature elements of all three research areas. More specifically, Paper I (p. 45) is concerned with a limitation of the sparse matrix algebra R package *spam* and modeling of large NDVI dataset. Paper III (p. 79) shows investigations of a distribution-free model for the prediction of missing values in satellite datasets and applications to Arctic NDVI data. Paper IV (p. 97) elaborates on both software implementations and validation strategies for Bayesian hierarchical models on the basis of data applications. The papers II and V contribute to two of the three fields. Paper II (p. 59) is concerned with R interfaces to compiled code and limitations for large data objects. Paper V (p. 131) presents statistical analyses of Arctic vegetation measurements using established computational methods.

4 Synthesis

This PhD thesis presents cutting-edge research in the field of applied statistics with a focus on Arctic ecology. The research questions stated in Section 3 are answered in the Papers I to V. To address the research question R1 we developed the R package *spam64*, which is an add-on package for *spam*. The package *spam64* enables handling sparse matrices with more than $2^{31} - 1$ non-zero elements and is currently the only sparse matrix algebra R package with that capability. The package also enabled fitting a likelihood based Gaussian process model to a large NDVI residual field; see also Subsection 2.1 for information on Gaussian process models. A more detailed description of the *spam64* related work is given in Paper I (p. 45). To implement *spam64* it was necessary to develop the R package *dotCall64*, which provides an enhanced R interface to compiled C/C++ and Fortran code. As a side project we answered R2 and present a technical description of *dotCall64* together with performance tests of the interface in Paper II (p. 59). The research question R3 was approached by a distribution-free model that is designed for parallel computing. The model was implemented in the R package *gapfill*, which is highlighted in Subsection 2.3. Paper III (p. 79) discusses the model and its performance using a NDVI dataset from the Arctic region. In comparison to two state-of-the-art methods the new gap-filling method provided the most accurate predictions. With respect to R4 we found that R in combination with the R package *spam* is an efficient way to implement Bayesian hierarchical models for spatial data. Graphical tools based on hierarchical clustering effectively characterized the Markov chains from different implementations. More information on the models and the comparison techniques are given in Paper IV (p. 97). Subsection 2.2 illustrates a Bayesian hierarchical model that is designed to combine Arctic field measurements and remotely sensed data. Last but not least, we investigated R5 using generalized mixed effects modes applied to summary statistics of field biodiversity observations and different remotely sensed data of the Arctic tundra. We could not establish significant relationships between these data types. Based on explanatory data analysis we showed that the partially missing data were a possible explanation for that finding. The analysis is discussed in Paper V (p. 131).

The statistical innovations developed in this thesis concern the three most common inference techniques, namely frequentist, Bayesian, and distribution-free models. The focus of the computational methods was put on large datasets, such as remote sensing based data products. Moreover, all the presented statistical and computational contributions are available in documented R packages together with instructive real-world data examples. This makes them not only interesting for computational statisticians, but also ready-to-use tools for applied scientists. Hence, all three scientific disciplines shown in Figure 6 are relevant to this thesis in *applied statistics*.

To proceed we see the following challenges and opportunities: The 64-bit extension of *spam* based on *dotCall64* worked well in the presented test cases (Papers I and II). However, increasing computing capabilities could motivate going a step further by defining all integers in R as 64-bit objects or by introducing a new 64-bit integer type. As this is a fundamental issue of the design of R, implementing a 64-bit integer type requires major changes in the source code. For user-friendly R implementations of gap-filling methods we see a great potential and interest based on the download numbers of the R package *gapfill* and questions from users of the package (Paper III). The implementation of *gapfill* could profit from the conversion of larger parts of the R code to the faster C++ language. Another extension of the method could be a model that jointly gap-fills multiple data layers from multivariate spatio-temporal satellite products. Concerning the validation of Bayesian hierarchical models it would be interesting to develop standardized testing procedures to make inference more trustworthy (Paper IV). Eventually, to enable combining Arctic field and remotely sensed data we recommend to use more unified sampling designs for field data and to retrieve and preprocess all recorded Landsat images (Paper V).

5 Bibliography

ACIA. Arctic Climate Impact Assessment: Scientific Report. Cambridge University Press, 2005. URL http://www.acia.uaf.edu.

Albert, J. Bayesian Computation with R. Springer, second edition edition, 2007. ISBN 9780387922980. doi:10.1007/978-0-387-92298-0.

Analytics, R. and Weston, S. *foreach*: Foreach looping construct for R, 2014. URL https://CRAN.R-project.org/package=foreach. R package version 1.4.2.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. Gaussian predictive process models for large spatial data sets. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70(4): 825–848, 2008. doi:10.1111/j.1467-9868.2008.00663.x.

Barnhart, K. R., Miller, C. R., Overeem, I., and Kay, J. E. Mapping the future expansion of Arctic open water. Nature Climate Change, 6(3):280–285, 2016. doi:10.1038/nclimate2848. Letter.

Berrocal, V. J., Gelfand, A. E., and Holland, D. M. Space-time data fusion under error in computer model output: An application to modeling air quality. Biometrics, 68(3):837–848, 2012. doi:10.1111/j.1541-0420.2011.01725.x.

Bevilacqua, M., Gaetan, C., Mateu, J., and Porcu, E. Estimating space and space-time covariance functions for large data sets: A weighted composite likelihood approach. Journal of the American Statistical Association, 107(497):268–280, 2012. doi:10.1080/01621459.2011.646928.

Bhatt, U. S., Walker, D. A., Raynolds, M. K., Comiso, J. C., Epstein, H. E., Jia, G., Gens, R., Pinzon, J. E., Tucker, C. J., Tweedie, C. E., and Webber, P. J. Circumpolar Arctic tundra vegetation change is linked to sea ice decline. Earth Interactions, 14(8):1–20, 2010. doi:10.1175/2010EI315.1.

Bi, J., Xu, L., Samanta, A., Zhu, Z., and Myneni, R. Divergent Arctic-boreal vegetation changes between North America and Eurasia over the past 30 years. Remote Sensing, 5(5):2093–2112, 2013. doi:10.3390/rs5052093.

Bliznyuk, N., Paciorek, C. J., Schwartz, J., and Coull, B. Nonlinear predictive latent process models for integrating spatio-temporal exposure data from multiple sources. The Annals of Applied Statistics, 8(3): 1538–1560, 9 2014. doi:10.1214/14-AOAS737.

Blok, D., Heijmans, M. M. P. D., Schaepman-Strub, G., Kononov, A. V., Maximov, T. C., and Berendse, F. Shrub expansion may reduce summer permafrost thaw in Siberian tundra. Global Change Biology, 16 (4):1296–1305, 2010. doi:10.1111/j.1365-2486.2009.02110.x.

Blok, D., Schaepman-Strub, G., Bartholomeus, H., Heijmans, M. M. P. D., Maximov, T. C., and Berendse, F. The response of Arctic vegetation to the summer climate: relation between shrub cover, NDVI, surface albedo and temperature. Environmental Research Letters, 6(3):035502, 2011. doi:10.1088/1748-9326/6/3/035502.

Bolin, D., Lindström, J., Eklundh, L., and Lindgren, F. Fast estimation of spatially dependent temporal vegetation trends using Gaussian Markov random fields. Computational Statistics & Data Analysis, 53 (8):2885–2896, 2009. doi:10.1016/j.csda.2008.09.017.

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., and White, J.-S. S. Generalized linear mixed models: a practical guide for ecology and evolution. Trends in Ecology & Evolution, 24(3):127–135, 2009. doi:10.1016/j.tree.2008.10.008.

Bolstad, W. M. Introduction to Bayesian Statistics. Wiley & Sons, Inc., second edition, 2007. ISBN 9780470181188. doi:10.1002/9780470181188.

Bradley, J. R., Cressie, N., and Shi, T. A comparison of spatial predictors when datasets could be very large. Statistics Surveys, 10:100–131, 2016. doi:10.1214/16-SS115.

Cadotte, M. W., Carscadden, K., and Mirotchnick, N. Beyond species: functional diversity and the maintenance of ecological processes and services. Journal of Applied Ecology, 48(5):1079–1087, 2011. doi:10.1111/j.1365-2664.2011.02048.x.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. Journal of Statistical Software, 76(1): 1–32, 2017. doi:10.18637/jss.v076.i01.

Chakraborty, A., De, S., Bowman, K. P., Sang, H., Genton, M. G., and Mallick, B. K. An adaptive spatial

model for precipitation data from multiple satellites over large regions. Statistics and Computing, 25(2): 389–405, 2015. doi:10.1007/s11222-013-9439-8.

Chapin, F. S., Sturm, M., Serreze, M. C., McFadden, J. P., Key, J. R., Lloyd, A. H., McGuire, A. D., Rupp, T. S., Lynch, A. H., Schimel, J. P., Beringer, J., Chapman, W. L., Epstein, H. E., Euskirchen, E. S., Hinzman, L. D., Jia, G., Ping, C.-L., Tape, K. D., Thompson, C. D. C., Walker, D. A., and Welker, J. M. Role of land-surface changes in Arctic summer warming. Science, 310(5748):657–660, 2005. doi:10.1126/science.1117368.

Chen, I.-C., Hill, J. K., Ohlemüller, R., Roy, D. B., and Thomas, C. D. Rapid range shifts of species associated with high levels of climate warming. Science, 333(6045):1024–1026, 2011a. doi:10.1126/science.1206432.

Chen, J., Zhu, X., Vogelmann, J. E., Gao, F., and Jin, S. A simple and effective method for filling gaps in Landsat ETM+ SLC-off images. Remote Sensing of Environment, 115(4):1053–1064, 2011b. doi:10.1016/j.rse.2010.12.010.

Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. ACM Transactions on Mathematical Software, 35 (3):22:1–22:14, 2008. doi:10.1145/1391989.1391995.

Colwell, R. K. Biodiversity: Concepts, Patterns, and Measurement, pages 257–263. Princeton University Press, 2009. doi:10.1515/9781400833023.257.

Cressie, N. and Wikle, C. Statistics for Spatio-Temporal Data. Wiley, 2015. ISBN 9781119243045.

Cressie, N. The origins of kriging. Mathematical Geology, 22(3):239–252, 1990. doi:10.1007/BF00889887.

Cressie, N. and Johannesson, G. Fixed rank kriging for very large spatial data sets. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70(1):209–226, 2008. doi:10.1111/j.1467-9868.2007.00633.x.

de Jong, R. Analysis of vegetation-activity trends in a global land degradation framework. 2012. ISBN 978-94-6173-312-2. Wageningen University.

de Jong, R., Schaepman, M. E., Furrer, R., de Bruin, S., and Verburg, P. H. Spatial relationship between climatologies and changes in global vegetation activity. Global Change Biology, 19(6):1953–1964, 2013. doi:10.1111/gcb.12193.

Deser, C., Walsh, J. E., and Timlin, M. S. Arctic sea ice variability in the context of recent atmospheric circulation trends. Journal of Climate, 13(3):617–633, 2000. doi:10.1175/1520-0442(2000)013<0617:ASIVIT>2.0.CO;2.

Didan, K., Munoz, A. B., Solano, R., and Huete, A. MODIS vegetation index user's guide (MOD13 Series), 2015. URL http://vip.arizona.edu/documents/MODIS/MODIS_VI_UsersGuide_June_2015_C6.pdf. Version 3.00 (Collection 6).

Eidsvik, J., Shaby, B. A., Reich, B. J., Wheeler, M., and Niemi, J. Estimation and prediction in spatial models with block composite likelihoods. Journal of Computational and Graphical Statistics, 23(2):295–315, 2014. doi:10.1080/10618600.2012.760460.

Eisenstat, S. C., Gursky, M. C., Schultz, M. H., and Sherman, A. H. Yale sparse matrix package I: The symmetric codes. International Journal for Numerical Methods in Engineering, 18(8):1145–1151, 1982. doi:10.1002/nme.1620180804.

Elmendorf, S. C., Henry, G. H. R., Hollister, R. D., Bjork, R. G., Boulanger-Lapointe, N., Cooper, E. J., Cornelissen, J. H. C., Day, T. A., Dorrepaal, E., Elumeeva, T. G., Gill, M., Gould, W. A., Harte, J., and et al. Plot-scale evidence of tundra vegetation change and links to recent summer warming. Nature Climate Change, 2(6):453–457, 2012a. doi:10.1038/nclimate1465.

Elmendorf, S. C., Henry, G. H. R., Hollister, R. D., Björk, R. G., Bjorkman, A. D., Callaghan, T. V., Collier, L. S., Cooper, E. J., Cornelissen, J. H. C., Day, T. A., Fosaa, A. M., Gould, W. A., Grétarsdóttir, J., Harte, J., Hermanutz, L., Hik, D. S., and others. Global assessment of experimental climate warming on tundra vegetation: heterogeneity over space and time. Ecology Letters, 15(2):164–175, 2012b. doi:10.1111/j.1461-0248.2011.01716.x.

Finazzi, F. and Fassò, A. D-STEM: A software for the analysis and mapping of environmental space-time variables. Journal of Statistical Software, 62(1):1–29, 2014. doi:10.18637/jss.v062.i06.

Finazzi, F., Scott, E. M., and Fassò, A. A model-based framework for air quality indices and population risk evaluation, with an application to the analysis of Scottish air quality data. Journal of the Royal Statistical Society: Series C (Applied Statistics), 62(2):287–308, 2013. doi:10.1111/rssc.12001.

Finley, A. O., Sang, H., Banerjee, S., and Gelfand, A. E. Improving the performance of predictive process modeling for large datasets. Computational Statistics & Data Analysis, 53(8):2873–2884, 2009. doi:10.1016/j.csda.2008.09.008.

Fuentes, M. Approximate likelihood for large irregularly spaced spatial data. Journal of the American Statistical Association, 102:321–331, 2007. doi:10.1198/016214506000000852.

Furrer, R. and Genton, M. G. Aggregation-cokriging for highly-multivariate spatial data. Biometrika, 98(3): 615–631, 2011. doi:10.1093/biomet/asr029.

Furrer, R. and Sain, S. R. *spam*: A Sparse Matrix R Package with Emphasis on MCMC Methods for Gaussian Markov Random Fields. Journal of Statistical Software, 36(10):1–25, 2010. doi:10.18637/jss.v036.i10.

Furrer, R., Genton, M. G., and Nychka, D. Covariance tapering for interpolation of large spatial datasets. Journal of Computational and Graphical Statistics, 15(3):502–523, 2006. doi:10.1198/106186006X132178.

Gaston, K. J. Global patterns in biodiversity. Nature, 405(6783):220–227, 2000. doi:10.1038/35012228.

Gentle, J. E. Matrix Algebra. Springer New York, 2007. ISBN 978-0-387-70872-0.

Gerber, F. *gapfill*: Fill missing values in satellite data, 2017. R package version 0.9.5-2.

Gilbert, J. R., Ng, E. G., and Peyton, B. W. An efficient algorithm to compute row and column counts for sparse Cholesky factorization. SIAM Journal on Matrix Analysis and Applications, 15:1075–1091, 1994.

Golub, G. H. and Loan, C. F. V. Matrix Computation. John Hopkins University Press, 1996. Third Edition.

Grace, J. B., Anderson, T. M., Olff, H., and Scheiner, S. M. On the specification of structural equation models for ecological systems. Ecological Monographs, 80(1):67–87, 2010. doi:10.1890/09-0464.1.

Gryparis, A., Coull, B. A., Schwartz, J., and Suh, H. H. Semiparametric latent variable regression models for spatiotemporal modelling of mobile source particles in the greater Boston area. Journal of the Royal Statistical Society: Series C (Applied Statistics), 56(2):183–209, 2007. doi:10.1111/j.1467-9876.2007.00573.x.

Hartman, L. and Hössjer, O. Fast kriging of large data sets with Gaussian Markov random fields. Computational Statistics & Data Analysis, 52(5):2331–2349, 2008. doi:10.1016/j.csda.2007.09.018.

Hennekens, S. M. and Schaminée, J. H. TURBOVEG, a comprehensive data base management system for vegetation data. Journal of Vegetation Science, 12(4):589–591, 2001. doi:10.2307/3237010.

Higdon, D. Space and space-time modeling using process convolutions. In Anderson, C. W., Barnett, V., Chatwin, P. C., and El-Shaarawi, A. H., editors, Quantitative Methods for Current Environmental Issues, pages 37–54. Springer, 2002. doi:10.1007/978-1-4471-0657-9_2.

Hill, M. O. Diversity and evenness: A unifying notation and its consequences. Ecology, 54(2):427–432, 1973. doi:10.2307/1934352.

Hinzman, L. D., Bettez, N. D., Bolton, W. R., Chapin, F. S., Dyurgerov, M. B., Fastie, C. L., Griffith, B., Hollister, R. D., Hope, A., Huntington, H. P., Jensen, A. M., Jia, G. J., Jorgenson, T., Kane, D. L., Klein, D. R., Kofinas, G., Lynch, A. H., Lloyd, A. H., McGuire, A. D., Nelson, F. E., Oechel, W. C., Osterkamp, T. E., Racine, C. H., Romanovsky, V. E., Stone, R. S., Stow, D. A., Sturm, M., Tweedie, C. E., Vourlitis, G. L., Walker, M. D., Walker, D. A., Webber, P. J., Welker, J. M., Winker, K. S., and Yoshikawa, K. Evidence and implications of recent climate change in northern Alaska and other Arctic regions. Climatic Change, 72(3):251–298, 2005. doi:10.1007/s10584-005-5352-2.

Hinzman, L. D., Deal, C. J., McGuire, A. D., Mernild, S. H., Polyakov, I. V., and Walsh, J. E. Trajectory of the Arctic as an integrated system. Ecological Applications, 23(8):1837–1868, 2013. doi:10.1890/11-1498.1.

Hmimina, G., Dufrêne, E., Pontailler, J.-Y., Delpierre, N., Aubinet, M., Caquet, B., de Grandcourt, A., Burban, B., Flechard, C., Granier, A., Gross, P., Heinesch, B., Longdoz, B., Moureaux, C., Ourcival, J.-M., Rambal, S., André, L. S., and Soudani, K. Evaluation of the potential of MODIS satellite data to predict vegetation phenology in different biomes: An investigation using ground-based NDVI measurements. Remote Sensing of Environment, 132:145–158, 2013. doi:10.1016/j.rse.2013.01.010. URL http://www.sciencedirect.com/science/article/pii/S0034425713000229.

Huete, A., Didan, K., Miura, T., Rodriguez, E., Gao, X., and Ferreira, L. Overview of the radiometric and biophysical performance of the MODIS vegetation indices. Remote Sensing of Environment, 83(1–2): 195–213, 2002. doi:10.1016/S0034-4257(02)00096-2.

IPCC. Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press,

Cambridge, United Kingdom and New York, NY, USA, 2013. ISBN 978-1-107-66182-0. doi:10.1017/CBO9781107415324. URL www.climatechange2013.org.

Jones, H. G. and Vaughan, R. A. Remote sensing of vegetation: principles, techniques, and applications. Oxford University Press, 2010. ISBN 978-0-19-920779-4.

Jones, T. G., Coops, N. C., Gergel, S. E., and Sharma, T. Employing measures of heterogeneity and an object-based approach to extrapolate tree species distribution data. Diversity, 6(3):396–414, 2014. doi:10.3390/d6030396.

Jost, L. Entropy and diversity. Oikos, 113(2):363–375, 2006. doi:10.1111/j.2006.0030-1299.14714.x.

Juszak, I., Eugster, W., Heijmans, M. M. P. D., and Schaepman-Strub, G. Contrasting radiation and soil heat fluxes in Arctic shrub and wet sedge tundra. Biogeosciences, 13(13):4049–4064, 2016. doi:10.5194/bg-13-4049-2016.

Juszak, I., Erb, A. M., Maximov, T. C., and Schaepman-Strub, G. Arctic shrub effects on NDVI, summer albedo and soil shading. Remote Sensing of Environment, 153:79–89, 2014. doi:10.1016/j.rse.2014.07.021.

Karl, T. R. and Trenberth, K. E. Modern global climate change. Science, 302(5651):1719–1723, 2003. doi:10.1126/science.1090228.

Korner-Nievergelt, F., Roth, T., Felten, S. v., Guélat, J., Almasi, B., and Korner-Nievergelt, P. Bayesian Data Analysis in Ecology Using Linear Models with R, BUGS, and STAN. Academic Press, 2015. ISBN 978-0-12-801370-0. doi:10.1016/B978-0-12-801370-0.01001-7.

Lang, S. I., Cornelissen, J. H. C., Shaver, G. R., Ahrens, M., Callaghan, T. V., Molau, U., Ter Braak, C. J. F., Hölzer, A., and Aerts, R. Arctic warming on two continents has consistent negative effects on lichen diversity and mixed effects on bryophyte diversity. Global Change Biology, 18(3):1096–1107, 2012. doi:10.1111/j.1365-2486.2011.02570.x.

Lemos, R. T. and Sansó, B. A spatio-temporal model for mean, anomaly, and trend fields of North Atlantic sea surface temperature. Journal of the American Statistical Association, 104(485):5–18, 2009. doi:10.1198/jasa.2009.0018.

Limpens, J., Berendse, F., Blodau, C., Canadell, J. G., Freeman, C., Holden, J., Roulet, N., Rydin, H., and Schaepman-Strub, G. Peatlands and the carbon cycle: from local processes to global implications—a synthesis. Biogeosciences, 5(5):1475–1491, 2008. doi:10.5194/bg-5-1475-2008.

Lindgren, F., Rue, H., and Lindström, J. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(4):423–498, 2011. doi:10.1111/j.1467-9868.2011.00777.x.

Loranty, M. M., Goetz, S. J., and Beck, P. S. A. Tundra vegetation effects on pan-Arctic albedo. Environmental Research Letters, 6(2):024014, 2011. doi:10.1088/1748-9326/6/2/024014.

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. The BUGS Book: A Practical Introduction to Bayesian Analysis. Texts in Statistical Science. Chapman & Hall/CRC, 2012. URL http://www.mrc-bsu.cam.ac.uk/bugs/thebugsbook/.

Madritch, M. D., Kingdon, C. C., Singh, A., Mock, K. E., Lindroth, R. L., and Townsend, P. A. Imaging spectroscopy links aspen genotype with below-ground processes at landscape scales. Philosophical Transactions of the Royal Society of London B: Biological Sciences, 369(1643), 2014. doi:10.1098/rstb.2013.0194.

Mamet, S. D., Young, N., Chun, K. P., and Johnstone, J. F. What is the most efficient and effective method for long-term monitoring of alpine tundra vegetation? Arctic Science, 2(3):127–141, 2016. doi:10.1139/as-2015-0020.

Mateu, J., Montes, F., and Fuentes, M. Recent advances in space-time statistics with applications to environmental data: An overview. Journal of Geophysical Research: Atmospheres, 108(D24), 2003. doi:10.1029/2003JD003819.

MPI. Message Passing Interface (MPI) forum, 2016. URL http://www.mpi-forum.org.

Myers-Smith, I. H., Forbes, B. C., Wilmking, M., Hallinger, M., Lantz, T., Blok, D., Tape, K. D., Macias-Fauria, M., Sass-Klaassen, U., Lévesque, E., Boudreau, S., Ropars, P., Hermanutz, L., Trant, A., Collier, L. S., Weijers, S., Rozema, J., Rayback, S. A., Schmidt, N. M., Schaepman-Strub, G., Wipf, S., Rixen, C., Ménard, C. B., Venn, S., Goetz, S., Andreu-Hayles, L., Elmendorf, S., Ravolainen, V., Welker, J., Grogan, P., Epstein, H. E., and Hik, D. S. Shrub expansion in tundra ecosystems: dynamics, impacts and research priorities. Environmental Research Letters, 6(4):045509, 2011. doi:10.1088/1748-9326/6/4/045509.

Myers-Smith, I. H., Elmendorf, S. C., Beck, P. S. A., Wilmking, M., Hallinger, M., Blok, D., Tape, K. D., Rayback, S. A., Macias-Fauria, M., Forbes, B. C., Speed, J. D. M., Boulanger-Lapointe, N., Rixen, C., Levesque, E., Schmidt, N. M., Baittinger, C., Trant, A. J., Hermanutz, L., Collier, L. S., Dawes, M. A., Lantz, T. C., Weijers, S., Jorgensen, R. H., Buchwal, A., Buras, A., Naito, A. T., Ravolainen, V., Schaepman-Strub, G., Wheeler, J. A., Wipf, S., Guay, K. C., Hik, D. S., and Vellend, M. Climate sensitivity of shrub growth across the tundra biome. Nature Climate Change, 5(9):887–891, 2015. doi:10.1038/nclimate2697. Letter.

Ng, E. G. and Peyton, B. W. Block sparse Cholesky algorithms on advanced uniprocessor computers. SIAM Journal on Scientific Computing, 14(5):1034–1056, 1993. doi:10.1137/0914063.

Nychka, D., Bandyopadhyay, S., Hammerling, D., Lindgren, F., and Sain, S. A multiresolution Gaussian process model for the analysis of large spatial datasets. Journal of Computational and Graphical Statistics, 24(2):579–599, 2015. doi:10.1080/10618600.2014.914946.

Oechel, W. C., Hastings, S. J., Vourlrtis, G., Jenkins, M., Riechers, G., and Grulke, N. Recent change of Arctic tundra ecosystems from a net carbon dioxide sink to a source. Nature, 361(6412):520–523, 1993. doi:10.1038/361520a0.

OpenMP. OpenMP application program interface, version 4.5, 2016. URL http://www.openmp.org. Architecture review board.

Pearson, R. G., Phillips, S. J., Loranty, M. M., Beck, P. S. A., Damoulas, T., Knight, S. J., and Goetz, S. J. Shifts in Arctic vegetation and associated feedbacks under climate change. Nature Climate Change, 3(7): 673–677, 2013. doi:10.1038/nclimate1858. Letter.

Pereira, H. M., Ferrier, S., Walters, M., Geller, G. N., Jongman, R. H. G., Scholes, R. J., Bruford, M. W., Brummitt, N., Butchart, S. H. M., Cardoso, A. C., Coops, N. C., Dulloo, E., Faith, D. P., Freyhof, J., Gregory, R. D., Heip, C., Höft, R., Hurtt, G., Jetz, W., Karp, D. S., McGeoch, M. A., Obura, D., Onoda, Y., Pettorelli, N., Reyers, B., Sayre, R., Scharlemann, J. P. W., Stuart, S. N., Turak, E., Walpole, M., and Wegmann, M. Essential biodiversity variables. Science, 339(6117):277–278, 2013. doi:10.1126/science.1229931.

Pinzon, J. E. and Tucker, C. J. A non-stationary 1981–2012 AVHRR NDVI3g time series. Remote Sensing, 6(8):6929–6960, 2014. doi:10.3390/rs6086929.

Plummer, M. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, 2003. Available at https://www.r-project.org/conferences/DSC-2003/Drafts/Plummer.pdf.

Pouliot, D., Latifovic, R., and Olthof, I. Trends in vegetation NDVI from 1 km AVHRR data over Canada for the period 1985–2006. International Journal of Remote Sensing, 30(1):149–168, 2009. doi:10.1080/01431160802302090.

Proença, V., Martin, L. J., Pereira, H. M., Fernandez, M., McRae, L., Belnap, J., Böhm, M., Brummitt, N., García-Moreno, J., Gregory, R. D., Honrado, J. P., Jürgens, N., Opige, M., Schmeller, D. S., Tiago, P., and van Swaay, C. A. Global biodiversity monitoring: From data sources to essential biodiversity variables. Biological Conservation, 2016. doi:10.1016/j.biocon.2016.07.014. In Press.

R. A Language and Environment for Statistical Computing. R Core Team, R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

Rahmstorf, S. A semi-empirical approach to projecting future sea-level rise. Science, 315(5810):368–370, 2007. doi:10.1126/science.1135456.

Raynolds, M. K., Comiso, J. C., Walker, D. A., and Verbyla, D. Relationship between satellite-derived land surface temperatures, Arctic vegetation types, and NDVI. Remote Sensing of Environment, 112(4): 1884–1894, 2008. doi:10.1016/j.rse.2007.09.008.

Robert, C. and Casella, G. Monte Carlo Statistical Methods. Texts in Statistics. Springer-Verlag, second edition, 2004. ISBN 978-1-4757-4145-2. doi:10.1007/978-1-4757-4145-2.

Rocchini, D., Balkenhol, N., Carter, G. A., Foody, G. M., Gillespie, T. W., He, K. S., Kark, S., Levin, N., Lucas, K., Luoto, M., Nagendra, H., Oldeland, J., Ricotta, C., Southworth, J., and Neteler, M. Remotely sensed spectral heterogeneity as a proxy of species diversity: Recent advances and open challenges. Ecological Informatics, 5(5):318–329, 2010. doi:10.1016/j.ecoinf.2010.06.001.

Roy, D. P., Borak, J. S., Devadiga, S., Wolfe, R. E., Zheng, M., and Descloitres, J. The MODIS Land product quality assessment approach. Remote Sensing of Environment, 83(1–2):62–76, 2002. doi:10.

1016/S0034-4257(02)00087-1. The Moderate Resolution Imaging Spectroradiometer (MODIS): a new generation of Land Surface Monitoring.

Rue, H. and Held, L. Gaussian Markov Random Fields: Theory and Applications, volume 104 of Monographs on Statistics and Applied Probability. Chapman & Hall, London, 2005.

Rue, H., Martino, S., and Chopin, N. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71(2):319–392, 2009.

Sahu, S. K., Gelfand, A. E., and Holland, D. M. High-resolution space-time ozone modeling for assessing trends. Journal of the American Statistical Association, 102(480):1221–1234, 2007. doi:10.1198/016214507000000031.

Sahu, S. K. and Bakar, K. S. Hierarchical Bayesian autoregressive models for large space-time data with applications to ozone concentration modelling. Applied Stochastic Models in Business and Industry, 28 (5):395–415, 2012. doi:10.1002/asmb.1951.

Schaepman-Strub, G., Schaepman, M. E., Painter, T. H., Dangel, S., and Martonchik, J. V. Reflectance quantities in optical remote sensing—definitions and case studies. Remote Sensing of Environment, 103 (1):27–42, 2006. doi:10.1016/j.rse.2006.03.002.

Schaepman-Strub, G., Schaepman, M. E., Martonchik, J., Painter, T., and Dangel, S. Radiometry and reflectance: From terminology concepts to measured quantities. Newbury Park, CA, USA: SAGE, 2009. doi:10.4135/9780857021052.n15.

Schaepman-Strub, G., Iturrate, M., and Furrer, R. Towards assessing biodiversity feedbacks to climate in the Arctic—future application of the AVA. In Arctic Vegetation Archive (AVA) Workshop, page 101, 2013. URL http://www.academia.edu/download/38266688/AVA_Proceedings_AVA_Krakow_Sept_2013.pdf.

Schliep, E. M. and Hoeting, J. A. Multilevel latent Gaussian process model for mixed discrete and continuous multivariate response data. Journal of Agricultural, Biological, and Environmental Statistics, 18(4):492–513, 2013. doi:10.1007/s13253-013-0136-z.

Serreze, M. C., Walsh, J. E., Chapin, F. S., Osterkamp, T., Dyurgerov, M., Romanovsky, V., Oechel, W. C., Morison, J., Zhang, T., and Barry, R. G. Observational evidence of recent change in the northern high-latitude environment. Climatic Change, 46(1):159–207, 2000. doi:10.1023/A:1005504031923.

Shannon, C. E. A mathematical theory of communication. The Bell System Technical Journal, 27(3): 379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

Sigrist, F., Künsch, H. R., and Stahel, W. A. A dynamic nonstationary spatio-temporal model for short term prediction of precipitation. The Annals of Applied Statistics, 6(4):1452–1477, 12 2012. doi:10.1214/12-AOAS564.

Simpson, E. Measurement of diversity. Nature, 163(688), 1949. doi:10.1038/163688a0.

Stein, M. L. A modeling approach for large spatial datasets. Journal of the Korean Statistical Society, 37 (1):3–10, 2008. doi:10.1016/j.jkss.2007.09.001.

Stein, M. L., Chi, Z., and Welty, L. J. Approximating likelihoods for large spatial data sets. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 66(2):275–296, 2004. doi:10.1046/j.1369-7412.2003.05512.x.

Stow, D. A., Hope, A., McGuire, D., Verbyla, D., Gamon, J., Huemmrich, F., Houston, S., Racine, C., Sturm, M., Tape, K., Hinzman, L., Yoshikawa, K., Tweedie, C., Noyle, B., Silapaswan, C., Douglas, D., Griffith, B., Jia, G., Epstein, H., Walker, D., Daeschner, S., Petersen, A., Zhou, L., and Myneni, R. Remote sensing of vegetation and land-cover change in Arctic Tundra Ecosystems. Remote Sensing of Environment, 89(3):281–308, 2004. doi:10.1016/j.rse.2003.10.018. URL http://www.sciencedirect.com/science/article/pii/S0034425703002803.

Stuart, A., Ord, K., and Arnold, S. Kendall's Advanced Theory of Statistics, Volume 2A, Classical Inference and the Linear Model. John Wiley & Sons, 2008. ISBN 978-0-470-68924-0.

Sturm, M., Douglas, T., Racine, C., and Liston, G. E. Changing snow and shrub conditions affect albedo with global implications. Journal of Geophysical Research: Biogeosciences, 110(G1), 2005. doi:10.1029/2005JG000013.

Sun, Y., Li, B., and Genton, M. Geostatistics for large datasets. In Porcu, E., Montero, J.-M., and Schlather, M., editors, Advances and Challenges in Space-time Modelling of Natural Events, volume 207

of Lecture Notes in Statistics, pages 55–77. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-17085-0. doi:10.1007/978-3-642-17086-7_3.

Swann, A. L., Fung, I. Y., Levis, S., Bonan, G. B., and Doney, S. C. Changes in Arctic vegetation amplify high-latitude warming through the greenhouse effect. Proceedings of the National Academy of Sciences, 107(4):1295–1300, 2010. doi:10.1073/pnas.0913846107.

Tjørve, E. and Tjørve, K. M. Species–Area Relationship. John Wiley & Sons, Ltd, 2001. ISBN 9780470015902. doi:10.1002/9780470015902.a0026330.

Tobler, W. A computer movie simulating urban growth in the Detroit region. Economic Geography, 46(2): 234–240, 1970.

Tucker, C. J., Pinzon, J. E., Brown, M. E., Slayback, D. A., Pak, E. W., Mahoney, R., Vermote, E. F., and Saleous, N. E. An extended AVHRR 8 km NDVI dataset compatible with MODIS and SPOT vegetation NDVI data. International Journal of Remote Sensing, 26(20):4485–4498, 2005. doi:10.1080/01431160500168686.

Tuomisto, H. A consistent terminology for quantifying species diversity? Yes, it does exist. Oecologia, 164 (4):853–860, 2010. doi:10.1007/s00442-010-1812-0.

Turner, D. P. Global vegetation monitoring: toward a sustainable technobiosphere. Frontiers in Ecology and the Environment, 9(2):111–116, 2011. doi:10.1890/090171.

Varin, C., Reid, N., and Firth, D. An overview of composite likelihood methods. Statistica Sinica, 21(1): 5–42, 2011. URL http://www.jstor.org/stable/24309261.

Vecchia, A. V. Estimation and model identification for continuous spatial processes. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 50(2):297–312, 1988. URL http://www.jstor.org/stable/2345768.

Virtanen, R., Grytnes, J.-A., Lenoir, J., Luoto, M., Oksanen, J., Oksanen, L., and Svenning, J.-C. Productivity–diversity patterns in Arctic tundra vegetation. Ecography, 36(3):331–341, 2013. doi:10.1111/j.1600-0587.2012.07903.x.

Wahba, G. Spline Models for Observational Data. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1990.

Walker, D. A., Epstein, H. E., Jia, G. J., Balser, A., Copass, C., Edwards, E. J., Gould, W. A., Hollingsworth, J., Knudson, J., Maier, H. A., Moody, A., and Raynolds, M. K. Phytomass, LAI, and NDVI in northern Alaska: Relationships to summer warmth, soil ph, plant functional types, and extrapolation to the circumpolar Arctic. Journal of Geophysical Research: Atmospheres, 108(D2), 2003. doi:10.1029/2001JD000986.

Walker, D., Breen, A., Druckenmiller, L., Wirth, L., Fisher, W., Raynolds, M., Sibík, J., Walker, M., and others. The Alaska Arctic Vegetation Archive (AVA-AK). Phytocoenologia, 46(2):221–229, 2016. doi:10.1127/phyto/2016/0128.

Walker, D. A., Raynolds, M. K., Daniels, F. J. A., Einarsson, E., Elvebakk, A., Gould, W. A., Katenin, A. E., Kholod, S. S., Markon, C. J., Melnikov, E. S., Moskalenko, N. G., Talbot, S. S., Yurtsev, B. A., Bliss, L. C., Edlund, S. A., Zoltai, S. C., Wilhelm, M., Bay, C., Gudjonsson, G., Ananjeva, G. V., Drozdov, D. S., Konchenko, L. A., Korostelev, Y. V., Ponomareva, O. E., Matveyeva, N. V., Safranova, I. N., Shelkunova, R., Polezhaev, A. N., Johansen, B. E., Maier, H. A., Murray, D. F., Fleming, M. D., Trahan, N. G., Charron, T. M., Lauritzen, S. M., and Vairin, B. A. The circumpolar Arctic vegetation map. 16 (3):267–282, 2005. doi:10.1111/j.1654-1103.2005.tb02365.x.

Weiss, D. J., Atkinson, P. M., Bhatt, S., Mappin, B., Hay, S. I., and Gething, P. W. An effective approach for gap-filling continental scale remotely sensed time-series. ISPRS Journal of Photogrammetry and Remote Sensing, 98:106–118, 2014. doi:10.1016/j.isprsjprs.2014.10.001.

Whittaker, R. H. Evolution and measurement of species diversity. Taxon, 21(2/3):213–251, 1972. doi:10.2307/1218190.

Wilson, A. M., Silander, J. A., Gelfand, A., and Glenn, J. H. Scaling up: linking field data and remote sensing with a hierarchical model. International Journal of Geographical Information Science, 25(3): 509–521, 2011. doi:10.1080/13658816.2010.522779.

Xu, G., Liang, F., and Genton, M. G. A Bayesian spatio-temporal geostatistical model with an auxiliary lattice for large datasets. Statistica Sinica, 25(1):61–79, 2015. doi:10.5705/ss.2013.085w.

Zimov, S. A., Schuur, E. A. G., and Chapin, F. S. Permafrost and the global carbon budget. Science, 312 (5780):1612–1613, 2006. doi:10.1126/science.1128908.

Paper overview

This thesis consists of five manuscripts. Their contents are briefly summarized in the following.

Paper I (p. 45)

Extending R Packages to Support 64-bit Compiled Code: An Illustration with spam64 and GIMMS $NDVI_{3g}$ Data

by Florian Gerber, Kaspar Mösinger & Reinhard Furrer

Abstract: Software packages for spatial data often implement a hybrid approach of interpreted and compiled programming languages. The compiled parts are usually written in C/C++ or Fortran, and are efficient in terms of computational speed and memory usage. Conversely, the interpreted part serves as a convenient user-interface and calls the compiled code for computationally demanding operations. The price paid for the user friendliness of the interpreted component is—besides performance—the limited access to low level and optimized code. An example of such a restriction is the 64-bit vector support of the widely used statistical language R. On the R side, users do not need to change existing code and may not even notice the extension. On the other hand, interfacing 64-bit compiled code efficiently is challenging. Since many R packages for spatial data could benefit from 64-bit vectors, we investigate strategies to efficiently pass 64-bit vectors to compiled languages. More precisely, we show how to simply extend existing R packages using the foreign function interface to seamlessly support 64-bit vectors. This extension is shown with the sparse matrix algebra R package *spam*. The new capabilities are illustrated with an example of GIMMS $NDVI_{3g}$ data featuring a parametric modeling approach for a non-stationary covariance matrix.

Scientific contributions: We present a strategy to link R packages with 32-bit and 64-bit compiled C/C++ and Fortran code using an enhanced foreign function interface. The advantages of our approach are that it is computationally efficient as it only uses the 64-bit compiled code if necessary and requires little additional efforts for developers. This new capabilities are illustrated with the sparse matrix algebra R package *spam*, which we used to fit a spatial model to a large GIMMS $NDVI_{3g}$ residual field. The Gaussian likelihood model implements a new approach to incorporate non-stationarity by letting the covariance depend on additional covariate fields, namely a digital elevation model and the distance to the nearest coast.

Authors contributions: Florian Gerber implemented the spatial model and fitted it to the GIMMS $NDVI_{3g}$ data. He contributed to the development of the 64-bit extension of the R package *spam* and wrote the manuscript. Kaspar Mösinger developed large part of the software and commented on the manuscript. Reinhard Furrer provided input on the statistical methodology, software development, and several versions of the manuscript.

Paper II (p. 59)

dotCall64: An Efficient Interface to Compiled C/C++ and Fortran Code Supporting Long Vectors

by Florian Gerber, Kaspar Mösinger & Reinhard Furrer

Abstract: The R functions `.C()` and `.Fortran()` can be used to call compiled C/C++ and Fortran code from R. This so-called foreign function interface is convenient, since it does not require any interactions with the C API of R. However, it does not support long vectors (i.e., vectors of more than 2^{31} elements). To overcome this limitation, the R package *dotCall64* provides `.C64()`, which can be used to call compiled C/C++ and Fortran functions. It transparently supports long vectors and does the necessary castings to pass numeric R vectors to 64-bit integer arguments of the compiled code. Moreover, `.C64()` features a mechanism to avoid unnecessary copies of function arguments, making it efficient in terms of speed and memory usage.

Scientific contributions: We describe a convenient way of using the open-source programming language R as a front-end for compiled C/C++ and Fortran code featuring 64-bit vectors. To that end, we introduce the R package *dotCall64* and provide technical details of its implementation as well as examples of how R code can be extended with 64-bit compiled C/C++ and Fortran code. The developed software is available at https://CRAN.R-project.org/package=dotCall64 and comes up with new opportunities for many R applications dealing with large data structures.

Authors contributions: Florian Gerber improved the R package *dotCall64* with unit tests and minor modifications of the code. He wrote the manuscript and parts of the documentation of the R package. Kaspar Mösinger developed large part of the software and commented on draft versions of the manuscript. Reinhard Furrer supervised the project and provided input on the manuscript.

Paper III (p. 79)

Predicting Missing Values in Spatio-Temporal Satellite Data

by Florian Gerber, Rogier de Jong, Michael E. Schaepman, Gabriela Schaepman-Strub & Reinhard Furrer

Abstract: Time series of remotely sensed optical data often contain data points of low product quality, related to atmospheric contamination or angular configuration for example. After detecting and removing such data points, the resulting data product is sparse and contains so called missing values. This is problematic for applications and signal processing methods that require temporally continuous data sets. To address this sparsity, we present a new gap filling method. We predict each missing value separately based on data points in the spatio-temporal neighborhood around the missing data point. The prediction of the missing values and the estimation of the corresponding prediction uncertainties are based on sorting algorithms and quantile regression. The gap filling method was applied to MODIS NDVI data from Alaska and tested with realistic scenarios featuring between 20% and 50% missing data. Validation against established methods showed that the proposed method has a good performance in terms of the root mean squared prediction error, which was between 0.041 and 0.060 and lower compared to the others methods for all test scenarios (Wilcoxon tests, all p-values $< 10^{-15}$). The method is available in the open-source R package *gapfill*. We demonstrate its performance using a real data example and show how it can be tailored to specific data sets. The computational workload can be distributed among several computers, rendering the method applicable to large data sets. Due to the flexible software design, users can control and redesign relevant parts with little additional effort. This makes it an interesting tool for gap filling satellite data and for the future development of gap filling methods.

Scientific contributions: We describe a new method to predict missing values in space-time data available on a regular grid. The method combines a dynamic procedure to divide the data into smaller subsets, image sorting techniques, and quantile regression. The developed software consists of R and C++ code and is available in the R package *gapfill* https://CRAN.R-project.org/package=gapfill. Tests based on MODIS NDVI data showed that *gapfill* provides accurate predictions, also in comparison to other state-of-the-art software.

Authors contributions: Florian Gerber developed the statistical method and implemented it in the R package *gapfill*. He did the presented statistical analyses. He wrote the manuscript and the documentation of the R code. Rogier de Jong, Michael E. Schaepman, Gabriela Schaepman-Strub commented drafts of the manuscript. Reinhard Furrer commented on the statistical methodology and several drafts of the manuscript.

Paper IV (p. 97)

Pitfalls in the Implementation of Bayesian Hierarchical Modeling of Areal Count Data: Illustration Using BYM and Leroux Models

by Florian Gerber & Reinhard Furrer

Abstract: Areal count data are often affected by random fluctuations, which makes the identification of spatial pattern challenging. Example data types are disease counts and environmental metrics recorded per administrative unit. To identify spatial trends in such datasets the Besag-York-Mollié (BYM) and the Leuroux models are used. These are Bayesian hierarchical models, for which no closed form expression of the posterior distribution exists. Therefore, fitting such models to data is based on Markov chain Monte Carlo (MCMC) methods or approximation techniques. However, the actual software implementations are error-prone because of the model complexity and the impossibility to test the long code sections in small units. This paper presents annotated code implementing the BYM and the Leroux models and discusses technical variations thereof. Moreover, graphical tools to compare sample(r)s and various practical tips for their implementation are given. We observed surprisingly large variability between different implementation methods as well as between different MCMC runs of one method.

Scientific contributions: This work provides documented and reproducible code for the implementation of various BYM and Leroux models. More precisely, it provides R and openBUGS code to set up MCMC and integrated nested Laplace approximation methods. Moreover, graphical tools to compare different model fits are presented and implemented in the R package *spam*. They revealed a surprisingly large difference between estimates from the different implementations. This questions the reliability of state-of-the-art methods and urges the need for validation procedures for related software.

Authors contributions: The project started as a side project of the MSc thesis of Florian Gerber, which was concerned with disease mapping for cancer and worm infections datasets. In an early phase of his PhD, Florian Gerber extended the investigations to alternative BYM model implementations and Leroux models. He also wrote this article and contributed to the development of the R package *spam*. Reinhard Furrer provided methodological input and commented on several versions of the manuscript.

Paper V (p. 131)

Challenges in Linking Arctic Plant Biodiversity with Satellite Based Landscape Characterizations

by Florian Gerber, Reinhard Furrer & Gabriela Schaepman-Strub

Abstract: Climate warming triggers changes in the Arctic vegetation, which in turn feedback to global climate. Therefore, an improved quantification of the Arctic vegetation and changes thereof could lead to reduced uncertainties in climate predictions. However, quantifying vegetation from the available field measurements is challenging because they are very sparse in space and time. One approach to nevertheless obtain a complete description is to extrapolate (upscale) the field measurements with the help of remote sensing based landscape characterizations. While several statistical methods are suitable for that task, one necessary prerequisite for all upscaling methods is the existence of a correlation pattern between the field measurements and the remote sensing based landscape characterizations at locations where both data types are available. In the presented study we tried to establish such a correlation pattern for biodiversity field measurements. We used a synthesis of plant abundance field measurements available from the International Tundra EXperiment (ITEX), which we then transformed into species richness and evenness biodiversity indices. For the landscape characterization we used spectral Landsat 5 and 7 satellite data as well as the ASTER global digital elevation model (GDEM). Both types of satellite data were used to characterize the landscapes by extracting the mean and/or the standard deviation of values near locations with field measurements. As the radius of the used discs is not clear a priori, we examined different radii ranging from 100 m to 10 km. The results of our analyses suggest that the extracted summary statistics from the Landsat data and the ASTER elevation model do not correlate with the species richness and evenness indices derived from the ITEX data.

Scientific contributions: The study contributes to a better understanding of Arctic field data and their relations to satellite based landscape characterizations. In particular, the study shows that the biodiversity indices from the considered field data are not correlated with various summary statistics of Landsat and ASTER data at several spatial scales. The concluding discussion points to possible issues with the data and is intended to contribute towards the goal of upscaling biodiversity estimates from Arctic field measurements meaningfully.

Authors contributions: Florian Gerber performed the analyses and wrote the manuscript. Gabriela Schaepman-Strub and Reinhard Furrer supported the data analysis with comments and helped with the choice of the considered datasets. They commented on several versions of the final report.

PAPER I

Extending R packages to support 64-bit compiled code: An illustration with spam64 and GIMMS $NDVI_{3g}$ data

Florian Gerber, Kaspar Mösinger & Reinhard Furrer

Paper published in the *Computers & Geoscience*

doi:10.18637/jss.v063.c01

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Extending R packages to support 64-bit compiled code: An illustration with spam64 and GIMMS $NDVI_{3g}$ data

Florian Gerber[a], Kaspar Mösinger[a], Reinhard Furrer[b,*]

[a] Department of Mathematics, University of Zurich, Switzerland
[b] Departments of Mathematics and Computational Science, University of Zurich, Switzerland

ARTICLE INFO

Keywords:
Sparse matrix
Non-stationarity
Compactly supported covariance function
Huge dataset
dotCall64
Foreign function interface

ABSTRACT

Software packages for spatial data often implement a hybrid approach of interpreted and compiled programming languages. The compiled parts are usually written in C, C++, or Fortran, and are efficient in terms of computational speed and memory usage. Conversely, the interpreted part serves as a convenient user-interface and calls the compiled code for computationally demanding operations. The price paid for the user friendliness of the interpreted component is—besides performance—the limited access to low level and optimized code. An example of such a restriction is the 64-bit vector support of the widely used statistical language R. On the R side, users do not need to change existing code and may not even notice the extension. On the other hand, interfacing 64-bit compiled code efficiently is challenging. Since many R packages for spatial data could benefit from 64-bit vectors, we investigate strategies to efficiently pass 64-bit vectors to compiled languages. More precisely, we show how to simply extend existing R packages using the foreign function interface to seamlessly support 64-bit vectors. This extension is shown with the sparse matrix algebra R package `spam`. The new capabilities are illustrated with an example of GIMMS $NDVI_{3g}$ data featuring a parametric modeling approach for a non-stationary covariance matrix.

1. Introduction

This research addresses the handling of very large vectors in R, and in our case was motivated through huge covariance matrices resulting from dependencies of georeferenced data, but any other scientific domain handling very large datasets could have served as motivation.

Spatial statistics relies on modeling the first *and* second order structures of directly observed or latent spatial fields. Typically, only one realization of such a spatial field is observed, and therefore parametric models for the second order structure is the prime choice. For maximum likelihood estimation or prediction (through classical kriging or other means) the covariance matrix of the spatial field has to be explicated. While the construction thereof is typically feasible, the operations based on these matrices (solving linear systems and calculating determinants) are the computational bottlenecks. Dataset sizes that can be dealt with a brute force implementation are on the order of several thousands—essentially the same size as a decade ago. However, with a careful model design, it is now possible to handle spatial datasets on the order of 10^5 to 10^6 on typical computing machines. These approaches can be classified into roughly two different categories. A model for which an efficient implementation exists (Kronecker formulation, separable models, e.g., Genton, 2007; Furrer and Genton, 2011) or for which an approximation is available (tapering, e.g., Furrer et al., 2006, Kaufman et al., 2008; Furrer et al., 2016; low-rank models, e.g., Cressie and Johannesson, 2008; Banerjee et al., 2008; Stein, 2008, composite likelihood approaches, e.g., Stein et al., 2004; Bevilacqua et al., 2012; Eidsvik et al., 2014, Gaussian Markov random fields type approximations, e.g., Hartman and Hössjer, 2008; Lindgren et al., 2011, etc.). For a review of statistical approaches for large datasets, see Sun et al. (2012).

While datasets on the order of 10^5 to 10^6 seem large, they are still much smaller than a typical Landsat 7 satellite image, which consists of more than 34 million pixels (30 m resolution for an approximate scene size of 170 km×183 km; source landsat.usgs.gov). Fitting spatial models on this data is challenging and limited by the available computing resources. Surprisingly the limiting factors are in some cases RAM and not the performance of the CPU(s). This is even more an issue when the computing software does not exploit the entire available memory. Until recently, the successful open source software R was bound to 32-bit addressing and thus limited the size of matrices independent of the available RAM. This limit implies that all (atomic) vectors have to have less than $2^{31} \approx 2.147 \cdot 10^9$ elements. At first sight

* Corresponding author.
E-mail address: reinhard.furrer@math.uzh.ch (R. Furrer).

http://dx.doi.org/10.1016/j.cageo.2016.11.015
Received 4 October 2015; Received in revised form 21 November 2016; Accepted 30 November 2016
Available online 07 December 2016

this seems huge, however a covariance matrix of a 160×320 lat/lon grid cannot be managed (corresponding to T106 spectral grid resolution climatedataguide.ucar.edu/climate-model-evaluation/common-spectral-model-grid-resolutions). Naturally, using sparse matrices (through, e.g., tapering) larger datasets are possible. However this limit is easily overdrawn with Landsat 7 satellite images and recommended taper ranges.

This article focuses on georeferenced data, but also the analysis of other data types is limited by the 32-bit constraint of R. One example from the authors recent research is the modeling of the covariance structure of microarray data with roughly 1.4 million probe sets on nowadays arrays https://www.affymetrix.com/catalog/131452/AFFY/Human+Exon+ST+Array, Furrer and Sain, 2009).

With the recent release of the R version 3.0.0, basic operations have been extended to be able to handle 64-bit vectors. However, it is not possible to directly pass long vectors from R to compiled code containing 64-bit integers though the foreign function interface. Although efforts exist to simplify the integration of compiled code in R (e.g., Eddelbuettel et al., 2016; Eddelbuettel, 2013), we are not aware of any interface that simplifies the interaction with 64-bit compiled code. This is unfortunate because there are many packages available for spatial data that relay on compiled code and could benefit from an extension to long vectors; for example, see the CRAN task views "Analysis of Spatial Data" (Bivand, 2016) and "Handling and Analyzing Spatio-Temporal Data" (Pebesma, 2016). This article sheds light on how to extend existing R packages with 64-bit compiled code and we will refer to such R packages as "64-bit packages." Since one can think of many possible approaches to cope with the 64-bit issue, we tried to find a strategy that has the following features: (i) From the end user perspective the enhanced 64-bit package should first of all cover all the functionality that was available before the extension without any performance losses in terms of memory usage and speed. Existing R code should be portable to the 64-bit package without any changes. Furthermore, the user should not be force to think about storage modes of vectors. (ii) From the developer perspective the work to migrate a package to 64-bit as well as the maintenance time should be kept at a minimum.

This article is structured as follows. Section 2 gives some background on how to call 64-bit compiled code from R. After covering general ideas and concepts, some technical details are given (a section that can be skipped). Finally, we introduce the R package `dotCall64` which simplifies the call to compiled code with 64-bit integers. Section 3 shows how to use both a 32-bit and a 64-bit version of the compiled code such that for small problems no computational and storage losses occur. Readers that are mainly interested in analyzing spatial data using huge sparse matrices are referred to Sections 4 and 5. In Section 4 we illustrate the porting of `spam`, an existing package to manage sparse matrices, to 64-bit capability and show the user relevant aspects with examples and performance measurements. In Section 5 we model the covariance of a GIMMS $NDVI_{3g}$ residual field involving "64-bit" Cholesky factors as a proof of concept. In Section 6 we conclude with a short discussion and outlook. The package `spam` and the R scripts that were used to create the figures and tables of this article are available at https://github.com/florafauna/CAGEO-spam64-supplement. A current development version of `spam` is available in the git repository https://git.math.uzh.ch/reinhard.furrer/spam.

2. Calling compiled code with 64-bit integers from R

2.1. General ideas and concepts

We now shed some light on the 64-bit implementation of R. While the focus of this section is the general concept, more technical insights are given in the next section. In R, vectors are one of the most basic object types. They can be thought of as a string of many elements that can be indexed according to their (relative) position. The indexing is based on (signed) 32-bit integers and thus the length of vectors is limited to $2^{31} - 1$ elements. Starting from release 3.0.0 in early 2013, basic support for vectors up to size 2^{52} is supplied; see also Section 12 on https://cran.r-project.org/doc/manuals/r-devel/R-ints.html. These vectors (including raw, logical, integer, numeric and character vectors, and lists and expression types) are called *long vectors*. The R implementation is such that for long vectors, doubles are used for addressing and minor modifications are required for the function `length()`, which returns a double in the case of long vectors. The extension has been done without breaking existing code and thus some of the implementation seems at first sight suboptimal. Notice that in R, matrices or general type arrays are objects where the data are stored in a vector and which possess a dimension attribute. Hence, the above-mentioned construction of long vectors still applies.

For efficient use, packages now have to supply the possibility of handling long vectors as well, and thus the underlying C/C++ or Fortran code has to be compiled in 64-bit mode. While the addressing in the compiled code is typically done with (signed) 64-bit integers, the discrepancy between the compiled and interpreted component are apparent.

There are several approaches to cope with this discrepancy in the storage mode and the two main ones are: (1) rewrite compiled code and use doubles instead of integers. (2) use doubles on the R side and cast them to 64-bit integers before calling the compiled code. The former requires a big effort on the package maintainers to rewrite existing code. Additionally, in the case R changes implementation to long integer addressing, many changes in the source code of the packages are required. The latter can be handled though an additional function that handles the type conversions (also denoted with casting) from double to 64-bit integers and back again. While this approach does not require any changes in existing compiled code, it implies a slight performance loss as casting between the storage modes takes time. We have evaluated the two approaches as well as additional flavors thereof (Mösinger, 2015) and chose the second approach.

Throughout the paper we will use the term "32-bit integers" to refer to the integer type in R and the 32-bit integers in the compiled code. On the other hand "64-bit integers" refer to doubles in R and 64-bit (long) integers in the compiled code.

2.2. Technical implementation

This section gives some technical insights into the underlying C implementation of the long vector support of R and can be skipped without loss of the general idea. We will refer to the R source code of version 3.1.0 (Core Team, 2016a) in several places by indicating the path to the source file and the line number. The mentioned files are available in the supplementary material of this paper. In addition, we highlight changes in the source files `src/include/Rinternals.h`, `src/include/Rinlinedfuns.h`, and `src/main/memory.c`, which were made to support long vectors. They are given in the files `diff_Rinternals.h`, `diff_Rinlinedfuns.h`, and `diff_memory.c` of the supplementary material and are the result of `svn diff -r 59004:59009` of the corresponding files in the R svn repository (http://svn.r-project.org/R/trunk/).

Long vectors have been introduced without breaking existing code. For example, the widely used C function `R_len_t length(SEXP s)` (defined in `src/include/Rinlinedfuns.h:122`) returns the length of a SEXP (S expression) as a `R_len_t`, which typedef'ed as `int32_t` (defined in `src/include/Rinternals.h:49`). Any code that assumes that `length(SEXP s)` returns an `int32_t` is compatible with this declaration. However, if `SEXP s` is a *long vector* and therefore the length cannot be stored inside an `int32_t`, the operation returns an error (See `src/main/memory.c:3828` which is called at `src/include/Rinternals.h:325`). Therefore, any legacy code that calls `R_len_t length(SEXP s)` still works on *short vectors* and does not need to be changed.

To get the length of *long vectors*, one has to call the newly defined function `R_xlen_t xlength(SEXP s)` (defined in `src/include/Rinlinedfuns.h:154`) instead. If R is compiled with *long vector* support, `R_xlen_t` is typedef'ed as `ptrdiff_t` (defined in `src/include/Rinternals.h:62`), where "`ptrdiff_t` is the signed integer type of the result of subtracting two pointers. This will probably

be one of the standard signed integer types (short int, int or long int), but might be a nonstandard type that exists only for this purpose." (The GNU C Library GNU, 2014, A.4 Important Data Types.)

We now sketch how *long vectors* are actually implemented. In R, vectors are made out of a header of type `VECSEXP` (defined in `src/include/Rinternals.h:273`) that is followed by the actual data. The header contains a field `length` of type `R_len_t` and hence cannot capture the length of a *long vector* as we have seen previously. Whenever the actual length is larger than $2^{31} - 1$, the length inside `VECSEXP` is set to -1 and an additional header of type `R_long_vec_hdr_t` is prefixed, which contains a field called `length` of type `R_xlen_t`.

The current implementation of R defines the R-type integer as signed 32-bit `int` (known as `int32_t`). There is no other integer type; there is no `R_xlen_t` equivalent in R. Instead, any integer number greater than $2^{31} - 1$ is stored as a double, which is integer-precise up to about 2^{52} (see `help("long vectors")`). As a consequence (1) *long vectors* are indexed by doubles and (2) if the length of a vector is larger than 2^{31} - 1 the R function `length(x)` returns a number of type double (see `help(length)`).

Note that the R package `bit64` (Oehlschlägel, 2015) provides a more efficient data type for 64-bit integers compared to the 64-bit integer provided by R. However, the package does not support *long vectors* and thus cannot be used in our context.

2.3. Alternative interface provided by the package `dotCall64`

Mösinger (2015) implemented an extension of R's foreign function interface, which is available in the R package `dotCall64` (Mösinger et al., 2016). The package provides an interface (written in C, exposed as an R function) that can be used to call compiled code in a way that (i) the arguments of the function are copied if and only if necessary (ii) 64-bit integers are cast from double (the R storage mode of 64-bit integers) to 64-bit integers before calling the compiled code and cast from 64-bit integers to doubles afterwards again (iii) supports long vectors.

Next, we illustrate how the `dotCall64` package can be used to call compiled code. Assume a hypothetical Fortran function `fun` that takes one integer argument `arg`. (The example would be similar for a C/C++ function.) Given the function is properly compiled and loaded in R, it can be called with the integer argument `arg=1L` via

```
R> out <- .Fortran("fun", arg)
```

The same call can be made via the R function `.C64()` from the R package `dotCall64`, which is available on CRAN.

```
R> install.packages("dotCall64")
R> library("dotCall64")
R> out <- .C64("fun", SIGNATURE = "int", arg)
```

Here, we additionally have to specify the argument `SIGNATURE`, which is set to integer in this case. In this situation, the result is the same as with the `.Fortran()` call. The main advantage of using `dotCall64` becomes obvious when `fun` is changed such that it takes a 64-bit integer as argument. If we now set `arg=2^32` and call the function via `.Fortran("fun", arg)` the Fortran code will interpret the 8 bytes of the double as a 64-bit integer, likely resulting in a crash. On the other hand, the call via `.C64()` can be adapted to expect a 64-bit integer by setting the argument `SIGNATURE` to `"int64"`. With that the call returns the desired result.

```
R> out <- .C64("fun", SIGNATURE = "int64", arg)
```

Instead of using the R function `.C64()`, the C function `dotCall64()` of the R package `dotCall64` can be called directly. This is especially useful when the compiled code relies on the C API of R or extensions thereof like, e.g., the R package `Rcpp` (Eddelbuettel et al., 2016; Eddelbuettel and François, 2011; Eddelbuettel, 2013). More detailed information on `dotCall64` including a description of the implementation and more extensive examples is given in Gerber et al. (2016).

3. Managing 32-bit and 64-bit compiled code

3.1. Motivation and general framework

As described in the previous section, the `dotCall64` interface takes care of the necessary casting when calling compiled code with 64-bit integer vectors from R. Hence, one could create an R package using doubles (instead of integers) on the R site and 64-bit integers in the compiled code. The drawback of this approach is that type conversion takes time and using 64-bit integers instead of integers is a waste of memory when the same could be done with integers. To illustrate this consider the extraction of one element out of a vector of length 2^{30} through compiled code. In the case where the vector is of type integer this operation is virtually instantaneous (order of milliseconds). On the other hand, if the same vector is stored as doubles (the format of a 64-bit integer in R) the same operation requires 4 Gb of additional physical RAM and takes about two seconds because of the necessity of casting from double to 64-bit integers. This motivates the uses of integers in R and calls compiled code with 32-bit integers whenever possible, or equivalently, uses the 64-bit variants only if necessary. Hence, the compiled code needs to be provided with both 32-bit integers and 64-bit integers.

We implemented the work flow of an R function calling potentially 64 bit compiled code as schematically illustrated in Fig. 1. The input of the R function may contain integers or 64-bit integer vectors, besides vectors of other types. After preprocessing the arguments in R, it is decided whether to use the compiled code with 32-bit integers or 64-bit integers. The compiled code is then called though a function from `dotCall64` that does the copying and casting of the arguments if necessary. Back in R, the results are post-processed and it is decided whether to returned the integer vectors as integers or doubles to R.

3.2. An S4 class to handle 32-bit and 64-bit integer vectors

As mentioned above it is beneficial in terms of performance to use the (32-bit) integer R type whenever possible to store integers, and doubles otherwise. This gets more involved when using S4 classes to

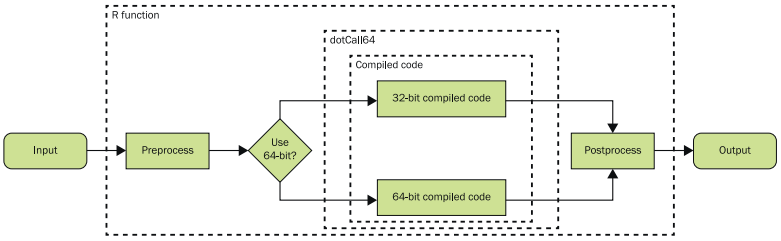


Fig. 1. Flow diagram of an R function calling compiled code as proposed in Section 3.1: First the arguments are preprocessed. Then it is decided whether to use the compiled code with 32-bit or 64-bit integers. Then the compiled code is called through functions provided by `dotCall64`, which takes care of copying and casting the arguments if required. In a post processing the step the results are collected and returned to `R`, having integers stored as integers or as doubles (64-bit integers).

store the data. To illustrate this we consider a class `simple` having one slot `entry` of type integer.

```
R> simple <- setClass(Class = "simple",
+    slots = c(entry = "integer"))
```

Since R stores 64-bit integer as doubles, this class cannot be used to store long vectors. Extending the class to accept long vectors efficiently while reassuring backward compatibility and maintainer friendliness is not trivial. Mösinger (2015) experimented with various designs. One idea is to let the class `simple` remain unchanged and define a new class `simple64` that is used to store 64-bit integers. In this situation one could, e.g., add an additional constructor function that decides whether to create an object of class `simple` or `simple64` or even overwrite the constructor of `simple` with that constructor. The disadvantage of this design is that we have to manage two S4 classes.

Another design that uses only one S4 class is illustrated with more details next. First, we modify the definition of the class `simple` to accept vectors of the class numeric.

```
R> simple <- setClass(Class = "simple",
+    slots = c(entry = "numeric"))
```

Note that the class numeric extends the class integer, and as a consequence, the slot `entry` accepts vectors of type double or integer. The decision to use one or the other type can be made in the initialize method of the class that we define as follows:

```
R> simple.init <- function(.Object, entry) {
+     if(max(abs(entry)) < .Machine$integer.max)
+          .Object@entry <- as.integer(entry)
+     else .Object@entry <- entry
+     return(.Object)
+ }
R> setMethod("initialize",
+    signature(.Object = "simple"),
+    simple.init)
[1] "initialize"
```

To illustrate the functionality of this class, we define the function `mult()`, which corresponds to a scalar multiplication of the class simple.

```
R> mult <- function(obj, factor)
+     return(simple(
+     entry = obj@entry * factor[1]))
```

Next, we create an instance of class simple:

```
R> print(s1 <- simple(entry = 2^29))
An object of class "simple"
Slot "entry":
[1] 536870912
```

Since 2^{29} is smaller than `.Machine$integer.max` (2^{31} in our environment) the slot `entry` is of type integer.

```
R> typeof(s1@entry)
[1] "integer"
```

When applying the function `mult()` to `s1` the appropriate storage format of the slot `entry` is chosen automatically.

```
R> s2 <- mult(s1, 8)
R> typeof(s2@entry)
[1] "double"
R> s3 <- mult(s2, 1/4)
R> typeof(s3@entry)
[1] "integer"
```

Note that the `initialize()` function is only called if the class constructor is called. Hence, the slot `entry` of the class can be overwritten without checking of the format via

```
R> s1@entry <- 1L
```

This may beneficial in terms of performance in some cases.

3.3. Code organization in two R packages

Now we have all necessary pieces together to extend an R package with 64-bit compiled code. At first sight, the organization of the code of such a package seems challenging from an R package developer point of view. Therefore, we give some insights in a code organization and development framework that reduces the additional work to a minimum.

Suppose we have a hypothetical package called `simplePkg` with 32-bit integers C/C++ or Fortran code called through the foreign function interface. To extend this package such that the functions can also be called with 64-bit integers we make use of the

Table 1
Distribution of code in two hypothetical R packages `simplePkg` and `simplePkg64`. The package `simplePkg` works with 32-bit compiled code and can be used independently of the other package. `simplePkg64` can be loaded as an add-on and enables the support for 64-bit vectors.

Package name	R code	Compiled code	manual
`simplePkg`	✓	✓(32-bit)	✓
`simplePkg64`	–	✓(64-bit)	–

Table 2
Code management in two hypothetical R packages `simplePkg` and `simplePkg64`. The first and third column summarize the basic file structure of the package. The middle column highlights the essential, minimal differences between the files, where <… indicates many more lines or files.

`simplePkg`		`simplePkg64`
`DESCRIPTION`	> `Package: simplePkg64` > `Depends: simplePkg` < `Package: simplePkg`	`DESCRIPTION`
`NAMESPACE`	> `useDynLib(simplePkg64)` < `useDynLib(simplePkg)` <…	`NAMESPACE`
`src`	Identical content for the source files. > The file `Makevars` containing (additional) compiler flags such as `PKG_FCFLAGS=-fdefault-integer-8`.	`src`
`man`	> Single file giving a short package overview <…	`man`
...	< More directories like `R`, `data`, `demo`, ...	

NAMESPACE feature of R. In R each package has a NAMESPACE, which allows the user to load different packages providing differently compiled functions with the same name. An argument in the foreign function interface (or in the dotCall64 functions) is then used to specify the package name and hence the compiled function is uniquely specified. This motivates the creation of an additional package simplePkg64 that contains the same source code for compiled code as in the package directory simplePkg/src/ but with 64-bit integers. This can be achieved with a simple call to GNU sed (2010) replacing all integer type declarations with "integer(8)". Alternatively, the GNU Fortran (2014) supports the flag -fdefault-integer-8, which can be set in simplePkg64/src/Makevars to declare integers as 64-bit integers. The remaining files and directories of the simplePkg64 are basically empty or reduced to a minimum. See Table 1 for a rough overview and Table 2 for a more detailed description of the (dis)similarities between the two packages. With this design, the package simplePkg works with 32-bit compiled code and can be used independently. Loading the simplePkg64 as an add-on enables the support of 64-bit integer vectors. We successfully tested the proposed strategy on Linux and Windows platforms.

Since the source code of the compiled code is the same in both packages, the additional time to maintain two packages instead of one is small. In fact, it is straightforward to design a Makefile that builds simplePkg and simplePkg64 out of one single package simplePkg.

4. Extending spam with 64-bit integer pointers

4.1. spam in a nutshell

We will now apply the ideas to the R package spam, which is an R package for sparse matrix algebra with emphasis on a Cholesky factorization of sparse positive definite matrices (Furrer and Sain, 2010). The implementation of spam is based on the competing philosophical maxims to be competitively fast compared to existing tools and to be easy to use, modify and extend. The first is addressed by using fast Fortran routines and the second by assuring S3 and S4 compatibility. One of the features of spam is to exploit the algorithmic steps of the Cholesky factorization and hence to perform only a fraction of the workload when factorizing matrices with the same sparsity structure. Simulations show that exploiting this break-down of the factorization results in a significant speed-up (Furrer and Sain, 2010).

To store the non-zero elements, spam essentially uses the "old Yale sparse format" (Eisenstat et al., 1977). In spam, a (sparse) matrix is stored as a S4 object with four slots (vectors), which are (1) the nonzero values row by row, (2) the ordered column indices of nonzero values, (3) the position in the previous two vectors corresponding to new rows, given as pointers, and (4) the column dimension of the matrix. Hence, to store a matrix with z nonzero elements spam requires z doubles and $z + n + 2$ integers compared to $n \times n$ doubles. Given the 32-bit limitation, we have the limit of (1) at most $2^{31} - 2$ rows, (2) at most $2^{31} - 1$ columns and (2) at most 2^{31} - 1 non-zero entries. More details about spam can be found in (Furrer and Sain, 2010; Gerber and Furrer, 2015).

4.2. Illustration of the functionality using spam64

The package spam64 is an implementation of the concept outlined in Section 3. It is based on spam version 2.x, which extends lower versions by modified Fortran calls and appropriate initializer methods. To enable 64-bit capability both the spam and the spam64 R packages need to be loaded (the latter depending on the former; see Table 2).

```
R> library("spam64")
R> grep(search(), pattern = "spam", value = TRUE)
[1] "package:spam"        "package:spam64"
```

As discussed above, the same top level R code for 32-bit and 64-bit matrices is used. Moreover, the user will notice the actual format only when looking explicitly at the storage format of the integer slots or by calling the print method of the spam object. As illustrated below, the functions return a spam object with 32-bit integers if possible and an object with 64-bit integers otherwise.

```
R> options(max.print = 14)
R> print(s1 <- spam(1:2^30))
Matrix of dimension 1073741824x1 with (row-wise)
nonzero elements:

[1]  1 2 3 4 5 6 7 8 9 10 11 12 13 14
[ reached getOption("max.print") -- omitted 1073741810
entries ]
Class 'spam' (32-bit)
R> print(s2 <- cbind(s1, s1))
Matrix of dimension 1073741824x2 with (row-wise)
nonzero elements:

[1] 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8
[ reached getOption("max.print") -- omitted 2147483633
entries ]
Class 'spam' (64-bit)
R> s2[1:2,]
     [,1] [,2]
[1,]    1    1
[2,]    2    2
Class 'spam' (32-bit)
```

In this R output, "(32-bit)" means that the slots colindices, rowpointers and dimension of the spam64 object are of type integer, as opposed to "(64-bit)" where these slots are of type double. The user may also force spam64 to return an object with slots of type double by setting the global option options(spam.force64=TRUE) or by setting the argument force64 of a specific function call to TRUE, e.g.,

```
R> spam(1, force64 = TRUE)
     [,1]
[1,]    1
Class 'spam' (64-bit)
```

4.3. Performance measurements

To get an impression of the performance in terms of speed and memory usage of spam64, we compared the implementation with spam and the matrix class from the base package using the following test setup: Matrices of dimension 2000×2000 with different percentages of randomly placed and randomly generated non-zero entries were generated. If the function to be tested required a positive definite matrix, this matrix was transformed into one with the same

amount of non-zero entries. Then, functions from the different implementations were applied to these matrices and their performance was measured. All elapsed CPU times reported in this manuscript were measured on a single Intel® Xeon® CPU E7-2850 at 2.00 GHz. To quantify the variability of the timing measures, the same function was applied 20 times on the same matrix. The memory usage was measured in terms of peak memory usage, i.e., the maximum amount of MB memory used during the calculations assessed with `gc()` and `gcTorture()`.

Since the measured matrices were relatively small, the `spam64` implementation would never switch to the 64-bit storage format in this setting. Therefore, additional measurements were taken with the `options(spam.force64=TRUE)`, where `spam64` uses the 64-bit storage format in any case.

In Fig. 2, some results for the matrix functions `t()` (transpose), `%*%` (matrix product), `cov()` (calculating the Wendland covariance matrix from a distance matrix) and `chol()` (Cholesky decomposition) are shown. We see that, first, the `spam64` 32-bit storage format has very similar results compared to the `spam` implementation. Second, the `spam64` 64-bit storage format adds a minor overhead because all pointer elements need to be cast from integers to double and vice versa. However, this casting can be easily distributed to multiple processors (task parallelization).

Classically known within the sparse matrix community yet possibly surprising for others, for many operations a significant amount of sparsity is needed to outperform the base implementation. The reduced amount of operations is offset by the handling of the storage structure. For example, replacing the first zero element by an arbitrary number is $O(z)$ for operation count. There is no overarching degree of sparsity when sparse matrices should be used. In addition to operation type, matrix size plays a role.

5. Non-stationary covariance model for a large NDVI residual field

Classical geostatistical models rely on parametric covariance functions to describe the spatial dependency structure of the date. Over the years, many models for anisotropic spatial processes have been proposed (see, e.g., Wackernagel, 2006). Such processes have a translation invariant covariance structure which can be parameterized with a few, typically with five to six, parameters. However, flexible non-stationary models are a quite recent research topic (Kleiber and Nychka, 2012). Most approaches proposed in the literature are very computing intensive and are not suitable for large spatial datasets.

The model proposed in Section 5.2 relaxes the stationarity assumption by allowing the covariance function to depend upon additional covariate data in a parametric way. The model is fitted to a satellite-based vegetation index using elevation data and the distance to the nearest coast as covariates. During the fitting procedure, the new capabilities of `spam64` are used and we get a grasp of the matrix sizes and their associated computation times when dealing with 64-bit integer vectors.

5.1. Data

The availability of long-term satellite earth observations enables the study of changes in the observations at large spatial extents. One primary variable of interest is the normalized difference vegetation

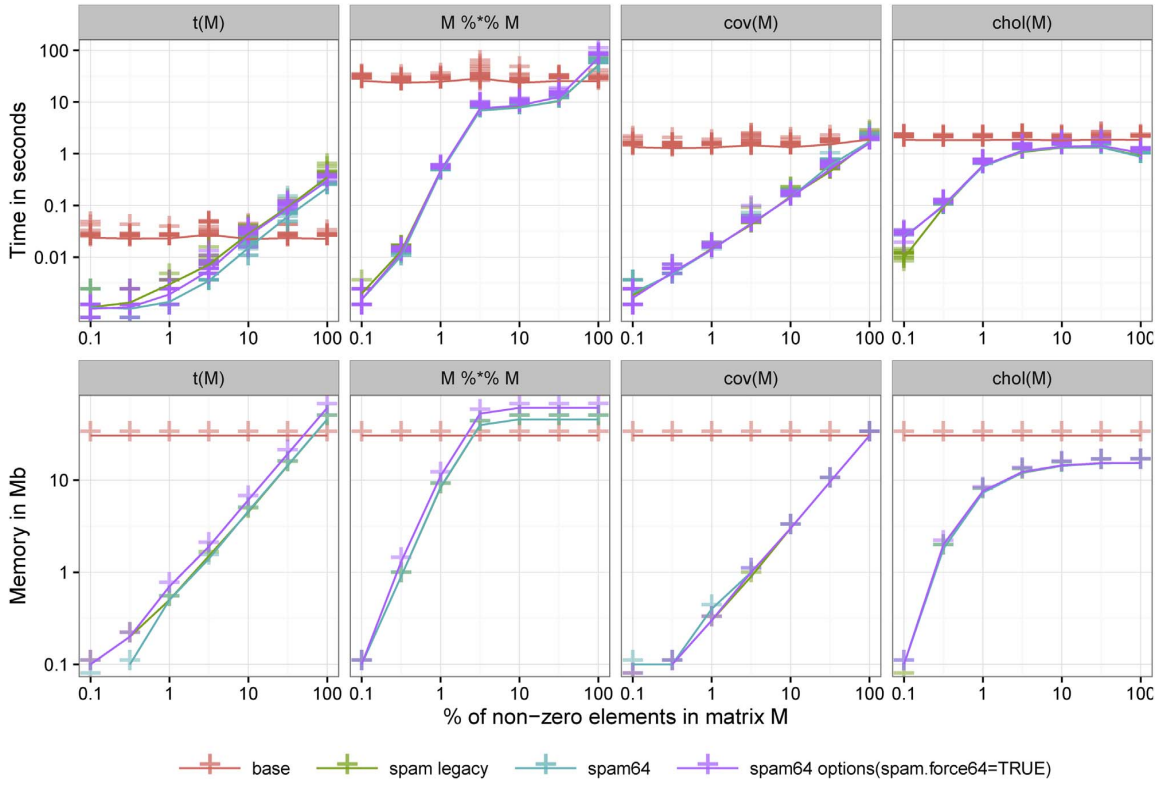


Fig. 2. Elapsed CPU time in seconds and peak memory in Mb for the matrix functions t(M) (transpose), M%*%M (matrix product), cov(M) (Wendland covariance function) and chol(m) (Cholesky decomposition). On the x-axis the % of non-zero elements of the 2000 × 2000 target matrix is indicated.

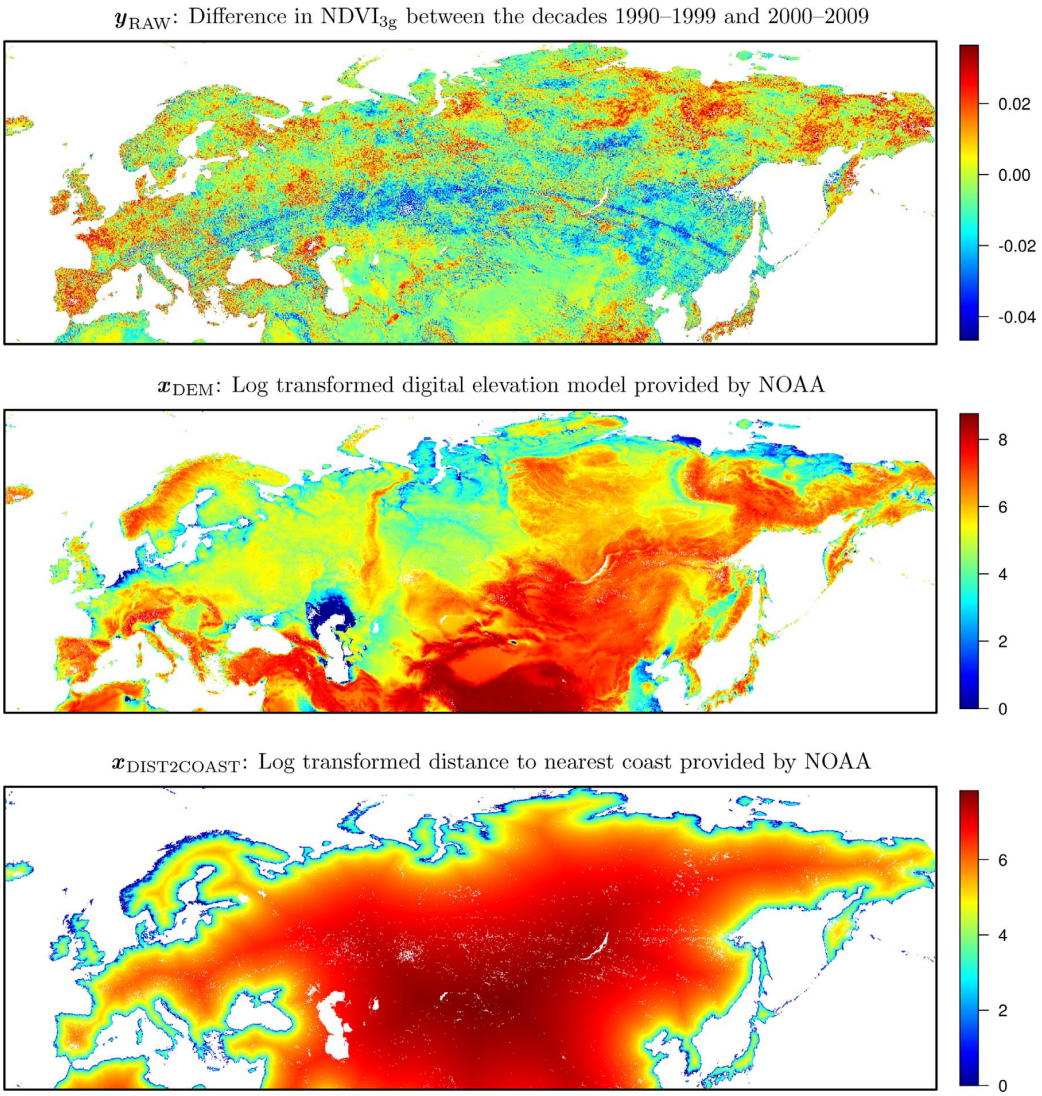


Fig. 3. Data used for the analysis. More precisely, we show y_{RAW} the $NDVI_{3g}$ difference field (top panel), x_{DEM} the log-transformed digital elevation model data (middle panel) and $x_{DIST2COAST}$ and the log transformed distance to the nearest coast (bottom panel).

index (NDVI), which serves as a proxy for the intensity of the vegetation through a normalized difference of the near infrared and the red color bands (Myneni and Hall, 1995). We consider the 3rd generation of the Global Inventory Monitoring and Modeling System (GIMMS) NDVI data product ($NDVI_{3g}$), which is a global time series of NDVI data based on information from different Advanced Very High Resolution Radiometer (AVHRR) satellites (Pinzon and Tucker, 2014). The product is available for the years 1981 to 2011 with a temporal resolution of 16 days and is provided on a regular latitude longitude grid with spatial resolution of 1/12° (≈8 km).

To investigate changes in the NDVI in Eurasia between the decades 1990–1999 and 2000–2009 the fields $y_{1990-1999}$ and $y_{2000-2009}$ were created by taking the corresponding 10-year average for each pixel. Then, the difference field was defined as $y_{RAW} = y_{2000-2009} - y_{1990-1999}$. Some pixels of y_{RAW} showed a large difference that was likely to reflect land cover changes from land to water and vice versa. The analysis of these large changes is of interest on its own, but given their large influence on the model fit they should be treated separately. Hence, the lower and the upper 1% quantile of y_{RAW} was removed, leaving n=769,940 observations for the analysis as shown in Fig. 3. To increase numerical stability the mean centered and scaled version of the difference field $y = (y_{RAW} - \bar{y}_{RAW})/\mathrm{sd}(y_{RAW})$ was finally modeled.

We construct the following covariates. First, starting from a 1 km elevation model provided by the National Oceanic and Atmospheric Administration (NOAA) (Hastings et al., 1999) we derived two spatial fields, namely, the logarithm of the elevation denoted as $\mathbf{x}_{DEM}$ and the logarithm of the variability of a 200 km box around the pixel denoted as $\mathbf{x}_{DEM\ VAR}$. Second, the distance to the nearest coast provided by the NOAA's National Ocean Service (oceancolor.gsfc.nasa.gov/DOCS/DistFromCoast/) was log-transformed and is denoted as $\mathbf{x}_{DIST2COAST}$. All three fields were resampled so that we have exactly one observation for each pixel of y. We define $\mathbf{X}$ as the matrix containing the columns $\mathbf{x}_{DEM}$, $\mathbf{x}_{DEM\ VAR}$ and $\mathbf{x}_{DIST2COAST}$. The data preparation and handling was greatly simplified by the R packages `rgdal` (Bivand et al., 2016), `raster` (Hijmans, 2016) and `sp` (Pebesma and Bivand, 2005; Bivand et al., 2013). Figures were made with `ggplot2` (Wickham, 2009) and

`fields` (Nychka et al., 2016).

5.2. Covariance model and implementation

We assume that the NDVI difference field $\boldsymbol{y}$ is a realization of a multivariate normal distribution with mean zero and covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. The covariance depends on the covariates and the parameters $\boldsymbol{\theta} = (\tau, \kappa, \boldsymbol{\beta})^\top$ and has the form

$$\boldsymbol{\Sigma}(\tau, \kappa, \boldsymbol{\beta}) = \kappa \mathbf{D}(\boldsymbol{\beta}) \mathbf{T}(\tau) \mathbf{D}(\boldsymbol{\beta}).$$

where $\kappa > 0$ is a scaling parameter. The diagonal matrix

$$\mathbf{D}(\boldsymbol{\beta}) = \mathrm{diag}(\exp(\mathbf{X}\boldsymbol{\beta})) \tag{1}$$

with $\boldsymbol{\beta} = (\beta_{\mathrm{DEM}}, \beta_{\mathrm{DIST2COAST}}, \beta_{\mathrm{DEM\ VAR}})^\top$ allows $\boldsymbol{\Sigma}$ to spatially vary according to the covariates. The matrix

$$\mathbf{T}(\tau) = (1 - \tau)\mathbf{I} + \tau \mathbf{R}, \quad \tau \in [0, 1]$$

is a weighted average between the identity matrix $\mathbf{I}$ and the correlation matrix $\mathbf{R}$. A similar decomposition was use by Leroux et al. (1999) for the inverse of the covariance matrix, whereas we use it for a substructure of the covariance matrix. In contrast to the classical signal to noise type decomposition, the parameter τ is bounded to the interval [0, 1], which is advantageous for the grid search optimization procedure used later. The matrix $\mathbf{R}$ was calculated via a Wendland (covariance) function (Wendland, 1995; Furrer et al., 2006), with a fixed range of 50 km using the great-circle distance of the spatial locations. Note that because of the use of the great-circle distance and the regular longitude/latitude grid of the data, the number of pixels included in the 50 km range vary with the latitude coordinate as illustrated in the lower panels of Fig. 5. The range parameter of the Wendland function could be estimated from the data, but is fixed here in order to increase the stability of the optimization procedure. The matrix $\mathbf{R}$ (and thus also $\mathbf{T}$) have about $1.28{\cdot}10^8$ non-zero entries (corresponding to about 0.02% of the entire matrix, see also Table 4).

We estimate the parameters $\boldsymbol{\theta}$ with maximum likelihood. Denoting $l(\boldsymbol{\theta}; \boldsymbol{y})$ the log-likelihood of the data we have:

$$-2l(\boldsymbol{\theta}; \boldsymbol{y}) = n \log(2\pi) + \log(\det(\boldsymbol{\Sigma}(\boldsymbol{\theta}))) + \boldsymbol{y}^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} \boldsymbol{y}. \tag{2}$$

The computationally expensive log-determinant and quadratic form are expressed as:

$$\log(\det(\boldsymbol{\Sigma})) = n \log(\kappa) + 2 \sum_{i=1}^{n} (\mathbf{X}\boldsymbol{\beta})_i + 2 \sum_{i=1}^{n} \log(\mathrm{chol}(\mathbf{T})_{ii}), \tag{3}$$

$$\boldsymbol{y}\boldsymbol{\Sigma}^{-1}\boldsymbol{y} = \mathbf{v}^\top \mathbf{T}^{-1} \mathbf{v}, \quad \text{where } \mathbf{v} = \mathbf{D}^{-1}\boldsymbol{y}/\sqrt{\kappa}. \tag{4}$$

where $(\mathbf{X}\boldsymbol{\beta})_i$ denoted the ith value of the vector, and $\mathrm{chol}(\mathbf{T})_{ii}$ denotes the ith diagonal entry of the Cholesky decomposition of $\mathbf{T}$. With this decomposition the most time-demanding calculation is the Cholesky decomposition of $\mathbf{T}$, which takes about 30 min on 12 Intel® Xeon® CPUs E7-2850 at 2.00 GHz (see also Table 4). To make the fitting procedure reasonably fast, a grid search was implemented for the parameter τ. More precisely, $\mathrm{chol}(\mathbf{T}(\tau))$ was calculated for $\tau \in S_\tau = \{0, 0.1, 0.2, \ldots, 1\}$ covering the entire range of the parameter space of τ. Then, $-2\,l(\kappa, \boldsymbol{\beta}; \boldsymbol{y}, \tau)$ was minimized for each value of $\tau \in S_\tau$ via the R function `optim()` using the option `method="L-BFGS-B"`, which calls a quasi-Newton optimizer allowing for box constraints (Byrd et al., 1995). Based on these results, a finer grid for τ with a spacing of 0.02 was defined as $S_\tau^{\mathrm{fine}} = \{0.52, 0.54, \ldots, 0.68\}$ and covered the most likely parameter range of τ according to the previous optimization results. Then, $-2\,l(\kappa, \boldsymbol{\beta}; \boldsymbol{y}, \tau)$ was minimized a second time for each value of $\tau \in S_\tau^{\mathrm{fine}}$. The left panel of Fig. 4 shows $\mathrm{argmax}_{\kappa,\boldsymbol{\beta}}\{l(\kappa, \boldsymbol{\beta}; \boldsymbol{y}, \tau)\}$ as a function of the evaluated values of τ. Finally, the value of τ that corresponds to the largest value of $\mathrm{argmax}_{\kappa,\boldsymbol{\beta}}\{l(\kappa, \boldsymbol{\beta}; \boldsymbol{y}, \tau)\}$, together with the configuration of κ and β are reported in Table 3.

For comparison, we also fit a stationary model by setting the diagonal matrix $\mathbf{D}$ to the identity matrix $\mathbf{I}$ in Eq. (1), i.e., $\boldsymbol{\beta} = \mathbf{0}$. Note that $\mathrm{chol}(\mathbf{T}(\tau))$ still depends on τ and therefore the two-step fitting procedure with a gird search for the parameter τ was used again.

5.3. Results and discussion of the model fit

For both covariance function models, the optimizer reported convergence for all values of τ. The estimated parameters and their uncertainty derived from the Hessian matrix are reported in Table 3. Note that due to the grid search for the parameter τ, its value was only estimated up to a resolution of 0.02 and the standard deviation cannot be derived directly. The log-likelihood as a function of the evaluated values of τ is shown for both models in the left panel of Fig. 4.

The estimated values $\boldsymbol{\beta}$ suggest that a smaller elevation and a small distance to the nearest coast result in a larger marginal variances. Furthermore, a small variance in the elevation seems to occur together with small covariance values of $\boldsymbol{y}$. This is in accordance with the observation that the NDVI is sensitive to the occurrence of water (Glenn et al., 2008; Friedl et al., 1995).

To further assess the fit of the models the diagonal elements of $\widehat{\boldsymbol{\Sigma}}$ were considered. For the model with a non-stationary covariance function, their distribution is shown as a histogram in the right panel of Fig. 4. The diagonal value of the stationary model is added as vertical line in the same plot. All values were reasonably close to 1, which is the true value of the variance if the data are modeled as independent observations. The diagonal elements of $\widehat{\boldsymbol{\Sigma}}$ of the non-stationary model are also shown on a map in Fig. 5, giving a visual impression of their spatial distribution. In the same

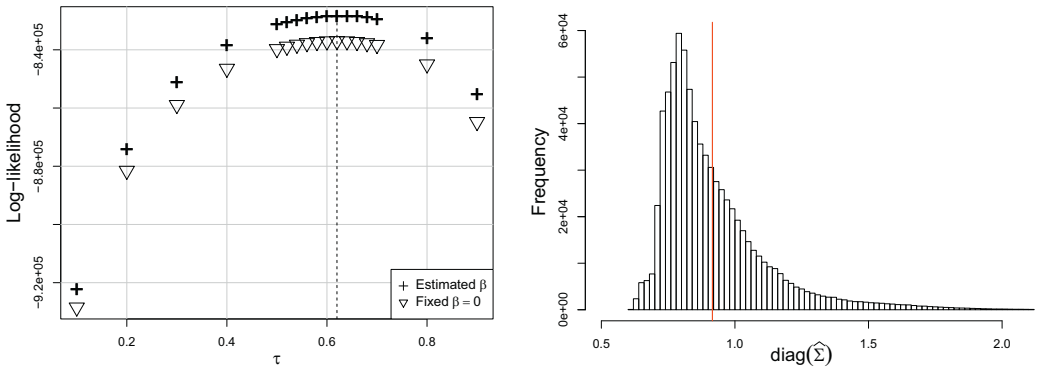


Fig. 4. (left) Log-likelihood for fixed values of τ both for the model with covariate data ($\boldsymbol{\beta}$ is estimated) and without covariate data ($\boldsymbol{\beta} = \mathbf{0}$). (right) Histogram of $\mathrm{diag}(\widehat{\boldsymbol{\Sigma}})$ for the model with covariate data (quartiles: 0.78, 0.86, 0.92 and 0.99). The corresponding value of the model without covariate data is indicated with the red vertical line.

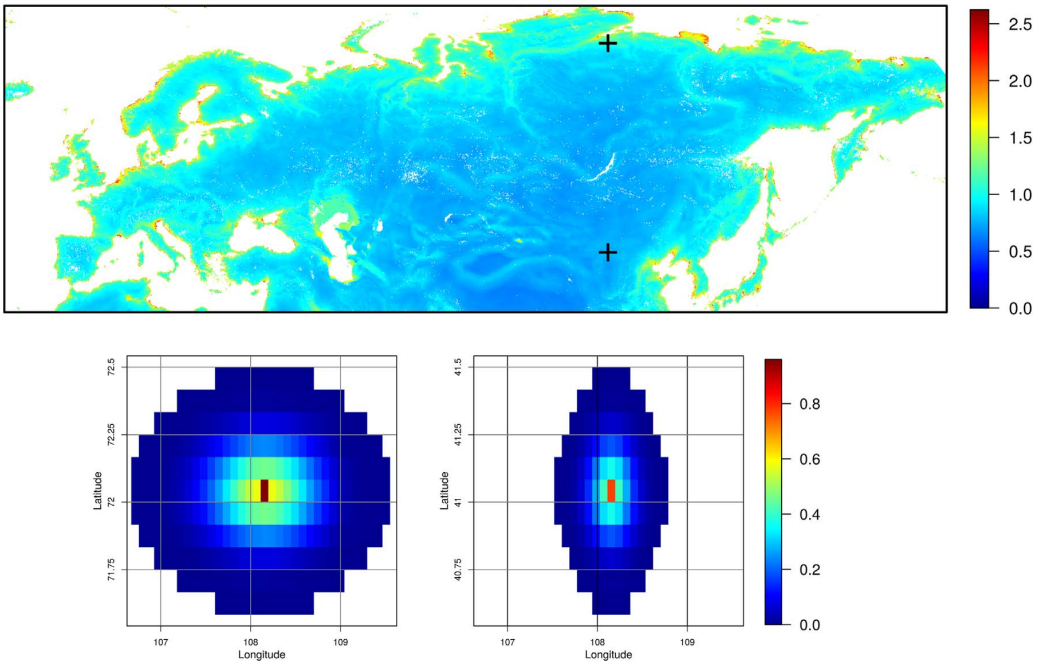


Fig. 5. Diagonal elements of the fitted covariance matrix, i.e., diag($\hat{\boldsymbol{\Sigma}}$) (top). For the two locations indicated with "+" the covariance with the surrounding pixels are shown in the lower panels. Here white areas indicate zero covariance due to its finite support (range) of 50 km. Since the data are stored on a regular latitude longitude grid and the great-circle distance was used, a circle with radius 50 km includes more pixels at the northern location ($N = 292$) compared to the southern location ($N = 121$).

Table 3
Estimates of the parameters τ, κ and $\boldsymbol{\beta}$ for both the non-stationary and the stationary model. For the parameters optimized with the quasi-Newton method the standard errors derived from the Hessian matrix are indicated in parenthesis.

	τ	κ	β_{DEM}	$\beta_{\text{DIST2COAST}}$	$\beta_{\text{DEM VAR}}$
Non-stationary	0.62	0.8973	−58.53	−54.38	36.05
		(0.0010)	(0.53)	(0.48)	(0.49)
Stationary	0.62	0.9157	0	0	0
		(0.0010)			

Table 4
Characteristics of the distance matrix **H**, correlation matrix **R** and Cholesky factor chol(**T**($\tau = 0.62$)) which were used to fit the covariance model in Section 5.2. The `spam` function name used to create the matrix together with the elapsed time, the size, and the density (percentages of non-zero elements) are given. The last column indicates whether 64-bit compiled code was used to generate the matrix.

Matrix	`spam` function	Elapsed time	Size	Density	64-bit
H	nearest.dist()	23.25 Minutes	1.4 Gb	0.02%	–
R	wendland.cov()	1.88 Minutes	1.4 Gb	0.02%	–
chol (**T**)	chol()	29.93 Minutes	8.5 Gb	0.19%	✓

figure the convariance structure for two spatial locations is plotted. It is worth noticing that in terms of the Bayesian information criteria, the more complex model with the non-stationary covariance function provided a better fit (BIC=1,658,282) compared to the model with a stationary covariance function (BIC=1,673,928).

Some characteristics of relevant matrices of the fitting procedure are indicated in Table 4. It is remarkable that only the Cholesky factor actually did use the 64-bit compiled code. However, such matrices can still be handled on a reasonably good desktop computer. From Table 4, we also see that it would take a considerable amount of time to optimize *all* parameters with `optim`, because every evaluation of the likelihood would then require a call to `chol()`. With the grid search approach for the parameter τ we did not only limit the total number of calls to `chol()` but also enabled parallel Cholesky decompositions and optimizations that reduced the fitting time to about 1.5 hours on 12 CPUs as specified above.

6. Discussion

We have illustrated a simple mechanism to extend R packages using the foreign function interface to 64-bit capability. The approach has two fundamental and advantageous benefits: (1) there is no computational overhead in terms of storage and time for small datasets from the end user; (2) the two-package solution is virtually maintenance free; (3) there are only a limited amount of changes in the R code of the original package required. These changes concern the S4 classes, such that these are capable of simultaneously, i.e., appropriately according to the vector length, handling the integers and double.

During this project, the testing phase of the software was disproportionately high. We started to test the functionality of `spam64` in a systematic way using the R package `testthat` (Wickham, 2016, 2011). With the two-package design the amount of tests basically doubles since all functions have to be tested with 32 and 64-bit integers. Actually using 64-bit integers in the testing procedure may increasing testing time quite a bit. However, it may be more safe to do so compared to shortcuts such as using 32-bit integers in conjunction with the `option(spam.force64=TRUE)`. Besides testing functions with respect to the correctness of their returned value, the performance in terms of memory usage and computation time is of great importance when dealing with large objects. Ideally, systematic performance tests in a framework similar to the unit testing are created. This allows the developer to monitor performance impacts of changes in the software

and supports the development of efficient software.

While this article focuses on a practical solution to increase the vector sizes that R can use in combination with compiled code, another aspect of manipulating large vectors is increasing the computation speed at which they are manipulated. The latter can be done by distributing the computational workload through, e.g., MPI (mpi-forum.org/) or OpenMP (www.openmp.org/) and is largely independent of the storage type of the vectors. We experimented with OpenMP to speedup the double to 64-integer castings done by `dotCall64` as described in Section 2. When using OpenMP in conjunction with R, the R package `OpenMPController` (Guest, 2013) was useful to control the number of threads from R. Besides the casting of vectors, there are some Fortran functions in `spam`/`spam64` that should be further optimized with a parallel implementation. Our current focus is the adoption of an efficient parallel Cholesky decomposition, which would enable us to fit the proposed non-stationary covariance model from Section 5 without a grid search for the parameter τ.

It is important to realize that working with huge matrices invokes a tremendous amount of computing time and we reckon that some users might be scared away. Therefore, it is worthwhile spending time install ing an optimized version of R, illustrated by the documentation of `help("long vectors")`: "For example on one particular platform chol on a 47,000 square matrix took about 5 h with the internal BLAS, 21 minutes using an optimized BLAS on one core, and 2 minutes using an optimized BLAS on 16 cores." More specifically, when installing R from its source code, options like `–disable-BLAS-shlib`, `–enable-R-profil-ing`, possibly `-O3` or similar for `CFLAGS` and `FFLAGS`, should be considered. The choice of the linear algebra package (e.g., BLAS (www.netlib.org/blas/), ATLAS (math-atlas.sourceforge.net/), ScaLAPACK (www.netlib.org/scalapack), MKL (software.intel.com/en-us/intel-mkl), openBLAS (www.openblas.net), SuperLU (http://crd-legacy.lbl.gov/~xiaoye/SuperLU/) is important as well. A related discussion can be found in Core Team (2016b).

In Section 2.3 we show an example where the foreign function interface `.Fortran()` is replaced by a call to `.C64()` from package `dotCall64`. The later imitates the `.Fortran()` style but is more efficient (Gerber et al., 2016). It is the authors' believe that the further development and support of a `.Fortran()` style interface is important since it simplifies the integration of compiled code that is not specifically tailored to R. Besides that it is quite popular: Among the 9,079 packages on CRAN (as of 2016-09-02), 12.9% (1,170 packages) make use of the foreign function interface. A comparable proportion use the modern interface to C/C++ (14.6%) and less than 2.1% use both approaches.

The storyline of the article was a spatial data analysis, which was chosen due to current research areas of the authors. We presented a non-stationary covariance model, which is an improvement over using isotropic covariance functions. Fitting the model to the chosen data required storing a Cholesky matrix with more than $2^{31} - 1$ non-zero elements. This was not possible with earlier versions of the R package `spam`, as the used foreign function interface of R does not support long vectors. We showed a way to extend `spam` to work with matrices having more than $2^{31} - 1$ non-zero elements. With that, handling the large Cholesky matrix used for the data analysis became feasible. The data example served as a solid proof of concept for the 64-bit extension strategy. In the area of "big data," there are seemingly countless occasions where manipulating huge vectors with compiled code is required and we are convinced that the move to 64-bit capability is a must that the R community has to address.

Acknowledgments

We thank two anonymous reviewers for their helpful comments. We acknowledge support of the University of Zurich Research Priority Program (URPP) on "Global Change and Biodiversity." We thank the NASA GIMMS team for providing the $NDVI_{3g}$ data for this analysis and Raphael Ostertag for providing the Python code to get the summary statistcs of the foreign function interfaces used by the R packages on CRAN. Furthermore, we thank Rogier de Jong for helpful discussions about the $NDVI_{3g}$ data.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2016.11.015.

References

Banerjee, S., Gelfand, A.E., Finley, A.O., Sang, H., 2008. Gaussian predictive process models for large spatial data sets. J. R. Statist. Soc. B 70, 825–848, http://doi.org/10.1111/j.1467-9868.2008.00663.x.

Bevilacqua, M., Gaetan, C., Mateu, J., Porcu, E., 2012. Estimating space and space-time covariance functions for large data sets: a weighted composite likelihood approach. J. Am. Statist. Assoc. 107, 268–280. http://dx.doi.org/10.1080/01621459.2011.646928.

Bivand, R.S., Pebesma, E., Gomez-Rubio, V., 2013. Applied Spatial Data Analysis with R, Second edition. Springer, NY. URL ⟨http://www.asdar-book.org/⟩.

Bivand, R., Keitt, T., Rowlingson, B., 2016. rgdal: Bindings for the Geospatial Data Abstraction Library. URL ⟨http://CRAN.R-project.org/package=rgdal⟩. R package version 1.1-10.

Bivand, R., 2016. CRAN task view: Analysis of spatial data. URL ⟨http://CRAN.R-project.org/view=Spatial⟩. version~2016-09-07.

Byrd, R.H., Lu, P., Nocedal, J., Zhu, C., 1995. A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. 16, 1190–1208. http://dx.doi.org/10.1137/0916069.

R Core Team, 2016a. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL ⟨http://www.R-project.org/⟩.

R Core Team, 2016b. R Installation and Administration. organization R Foundation for Statistical Computing. Vienna, Austria. URL ⟨https://cran.r-project.org/doc/manuals/R-admin.html⟩.

Cressie, N., Johannesson, G., 2008. Fixed rank kriging for very large spatial data sets. J. R. Statist. Soc. B 70, 209–226. http://dx.doi.org/10.1111/j.1467-9868.2007.00633.x.

Eddelbuettel, D., François, R., 2011. Rcpp: seamless R and C++ integration. J. Statist. Softw. 40, 1–18, URL ⟨http://www.jstatsoft.org/v40/i08/⟩.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Bates, D., Chambers, J., 2016. Rcpp: Seamless R and C++ Integration 2016.R package version 0.12.7. URL ⟨http://CRAN.R-project.org/package=Rcpp⟩. R package version 0.12.7.

Eddelbuettel, D., 2013. Seamless R and C++ Integration with Rcpp. Springer, New York, URL ⟨http://www.rcpp.org/book/⟩.

Eidsvik, J., Shaby, B.A., Reich, B.J., Wheeler, M., Niemi, J., 2014. Estimation and prediction in spatial models with block composite likelihoods. J. Comput. Graph. Stat. 23, 295–315. http://dx.doi.org/10.1080/10618600.2012.760460.

Eisenstat, S.C., Gursky, M.C., Schultz, M.H., Sherman, A.H., 1977. Yale sparse matrix package I: The symmetric codes. Research Report #112. Yale University, Department of Computer Science.

Friedl, M., Davis, F., Michaelsen, J., Moritz, M., 1995. Scaling and uncertainty in the relationship between the NDVI and land surface biophysical variables: an analysis using a scene simulation model and data from FIFE. Rem. Sens. Environ. 54, 233–246. http://dx.doi.org/10.1016/0034-4257(95)00156-5.

Furrer, R., Genton, M.G., 2011. Aggregation-cokriging for highly-multivariate spatial data. Biometrika 98, 615–631. http://dx.doi.org/10.1093/biomet/asr029.

Furrer, R., Sain, S.R., 2009. Spatial model fitting for large datasets with applications to climate and microarray problems. Stat. Comput. 19, 113–128. http://dx.doi.org/10.1007/s11222-008-9075-x.

Furrer, R., Sain, S.R., 2010. Spam: a sparse matrix R package with emphasis on MCMC methods for Gaussian Markov random fields. J. Stat. Softw. 36, 1–25. http://dx.doi.org/10.18637/jss.v036.i10.

Furrer, R., Genton, M.G., Nychka, D., 2006. Covariance tapering for interpolation of large spatial datasets. J. Comput. Graph. Stat. 15, 502–523, http://dx.doi.org/10.1198/106186006X132178.

Furrer, R., Bachoc, F., Du, J., 2016. Asymptotic properties of multivariate tapering for estimation and prediction. J. Multivariate Anal. 149, 177–191. http://dx.doi.org/10.1016/j.jmva.2016.04.006.

Genton, M.G., 2007. Separable approximations of space-time covariance matrices. Environmetrics 18, 681–695. http://dx.doi.org/10.1002/env.854.

Gerber, F., Furrer, R., 2015. Pitfalls in the implementation of Bayesian hierarchical modeling of areal count data: an illustration using BYM and Leroux models. J. Stat. Softw. 63, 1–32. http://dx.doi.org/10.18637/jss.v063.c01.

Gerber, F., Mösinger, K., Furrer, R., 2016. dotCall64: An efficient interface to compiled C/C++ and Fortran code supporting long vectors. submitted to the R journal.

Glenn, E.P., Huete, A.R., Nagler, P.L., Nelson, S.G., 2008. Relationship between remotely-sensed vegetation indices, canopy attributes and plant physiological processes: what vegetation indices can and cannot tell us about the landscape. Sensors 8, 2136. http://dx.doi.org/10.3390/s8042136.

GNU Fortran compiler, 2014. Reference manual for GCC version 4.9.2. URL ⟨http://gcc.

gnu.org/onlinedocs/gcc-4.9.2/gfortran/⟩.
GNU sed, 2010. Reference manual. URL ⟨http://www.gnu.org/software/sed/manual/⟩.
The GNU C Library, 2014. The GNU C library. URL ⟨http://www.gnu.org/software/libc/manual/html_node/index.html⟩.
Guest, S., 2013. OpenMPController: Control number of OpenMP threads dynamically. URL ⟨http://CRAN.R-project.org/package=OpenMPController⟩. R package version 0.1-2.
Hartman, L., Hössjer, O., 2008. Fast kriging of large data sets with Gaussian Markov random fields. Comput. Stat. Data Anal. 52, 2331–2349. http://dx.doi.org/10.1016/j.csda.2007.09.018.
Hastings, A., D., Dunbar, P.K., Elphingstone, G.M., Bootz, M., Murakami, H., Maruyama, H., Masaharu, H., Holland, P., Payne, J., Bryant, N.A., Logan, T.L., Muller, J.P., Schreierand, G., MacDonald, J.S., 1999. The Global Land One-kilometer Base Elevation (GLOBE) Digital Elevation Model, Version 1.0. National Oceanic and Atmospheric Administration, National Geophysical Data Center, 325 Broadway, Boulder, Colorado 80305-3328, U.S.A. Digital data base on the World Wide Web. URL ⟨http://www.ngdc.noaa.gov/mgg/topo/globe.html⟩.
Hijmans, R.J., 2016. raster: Geographic data analysis and modeling. URL ⟨http://CRAN.R-project.org/package=raster⟩. R package version 2.5-8.
Kaufman, C.G., Schervish, M.J., Nychka, D.W., 2008. Covariance tapering for likelihood-based estimation in large spatial data sets. J. Am. Stat. Assoc. 103, 1545–1555, http://dx.doi.org/10.1198/016214508000000959.
Kleiber, W., Nychka, D., 2012. Nonstationary modeling for multivariate spatial processes. J. Multivariate Anal. 112, 76–91. http://dx.doi.org/10.1016/j.jmva.2012.05.011.
Leroux, B.G., Lei, X., Breslow, N., 1999. Estimation of Disease Rates in Small Areas: A New Mixed Model for Spatial Dependence. IMA Volumes in Mathematics and its Applications, US Government Printing Office. http://dx.doi.org/10.1007/978-1-4612-1284-3_4.
Lindgren, F., Rue, H., Lindström, J., 2011. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. J. R. Stat. Soc. B 73, 423–498. http://dx.doi.org/10.1111/j.1467-9868.2011.00777.x.
Mösinger, K., Gerber, F., Furrer, R., 2016. dotCall64: Enhanced Foreign Function Interface Supporting Long Vectors. R package version 0.9-04. URL ⟨http://CRAN.R-project.org/package=dotCall64⟩.
Mösinger, K., 2015. An R implementation for huge spatiotemporal covariance matrices. Master's thesis. University of Zurich.
Myneni, R., Hall, F., 1995. The interpretation of spectral vegetation indexes. IEEE Trans. Geosci. Rem. Sens. 33, 481–486. http://dx.doi.org/10.1109/36.377948.
Nychka, D., Furrer, R., Paige, J., Sain, S., 2016. fields: Tools for Spatial Data. ⟨http://CRAN.R-project.org/package=fields⟩. R package version 8.4-1.
Oehlschlägel, J., 2015. bit64: A S3 class for vectors of 64bit integers. URL ⟨http://CRAN.R-project.org/package=bit64⟩. R package version 0.9-5.
Pebesma, E.J., Bivand, R.S., 2005. Classes and methods for spatial data in R. R News 5, 9–13.
Pebesma, E., 2016. CRAN task view: Handling and analyzing spatio-temporal data. URL ⟨http://CRAN.R-project.org/view=SpatioTemporal⟩. version 2016-08-18.
Pinzon, J.E., Tucker, C.J., 2014. A non-stationary 1981–2012 AVHRR $NDVI_{3g}$ time series. Remote Sensing 6, 6929. URL ⟨http://www.mdpi.com/2072-4292/6/8/6929⟩, http://dx.10.1111/10.3390/rs6086929.
Stein, M.L., Chi, Z., Welty, L.J., 2004. Approximating likelihoods for large spatial data sets. J. R. Stat. Soc. B 66, 275–296. http://dx.doi.org/10.1046/j.1369-7412.2003.05512.x.
Stein, M.L., 2008. A modeling approach for large spatial datasets. J. Korean Stat. Soc. 37, 3–10. http://dx.doi.org/10.1016/j.jkss.2007.09.001.
Sun, Y., Li, B., Genton, M., 2012. Geostatistics for large datasets, in: Porcu, E., Montero, J.M., Schlather, M. (Eds.), Advances and Challenges in Space-time Modelling of Natural Events. Springer Berlin Heidelberg. volume 207 of Lecture Notes in Statistics, pp. 55–77 https://dx.doi.org/10.1007/978-3-642-17086-7_3.
Wackernagel, H., 2006. Multivariate Geostatistics. third ed., Springer-Verlag, New York.
Wendland, H., 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math. 4, 389–396. http://dx.doi.org/10.1007/BF02123482.
Wickham, H., 2009. ggplot2: Elegant Graphics For Data Analysis. Springer, New York, URL ⟨http://www.ggplot2.org/book⟩.
Wickham, H., 2011. testthat: get started with testing. R J. 3, 5–10, URL ⟨https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf⟩.
Wickham, H., 2016. testthat: Unit Testing for R. URL ⟨https://CRAN.R-project.org/package=testthat⟩. R package version 1.0.2.

PAPER II

dotCall64: An Efficient Interface to Compiled C/C++ and Fortran Code Supporting Long Vectors

Florian Gerber, Kaspar Mösinger & Reinhard Furrer

Paper available on *arXiv*

Stable URL: https://arxiv.org/abs/1702.08188

dotCall64: An Efficient Interface to Compiled C/C++ and Fortran Code Supporting Long Vectors

by Florian Gerber, Kaspar Mösinger and Reinhard Furrer

Abstract The R functions `.C()` and `.Fortran()` can be used to call compiled C/C++ and Fortran code from R. This so-called foreign function interface is convenient, since it does not require any interactions with the C API of R. However, it does not support long vectors (i. e. , vectors of more than 2^{31} elements). To overcome this limitation, the R package **dotCall64** provides `.C64()`, which can be used to call compiled C/C++ and Fortran functions. It transparently supports long vectors and does the necessary castings to pass `numeric` R vectors to 64-bit integer arguments of the compiled code. Moreover, `.C64()` features a mechanism to avoid unnecessary copies of function arguments, making it efficient in terms of speed and memory usage.

Introduction

The interpreted character of R makes it a convenient front-end for a wide range of applications. Although R provides a rich infrastructure, it can be advantageous to extend R programs with compiled code written in C/C++ or Fortran (Eubank and Kupresanin, 2011). According to Chambers (2008), reasons for such an extension are the access to new and trusted computations, the increase in computational speed, and the object referencing capabilities. For completeness, we also list the reasons against such an extension, which include an increased workload to write, maintain, and debug the software, platform dependencies, and a less readable source code.

R provides two types of interfaces to call compiled code documented in "Writing R Extensions" (R Core Team, 2016a). First, the *modern interfaces to C/C++ code* feature the R functions `.Call()` and `.External()`. It enables accessing, modifying, and returning R objects from C/C++ using the C API of R (Wickham, 2014). On one hand, this is convenient when the C/C++ code is specifically written to be used with R. In that case, the C API serves as a glue between R and C/C++, providing some R functionality and control over copying R objects on the C/C++ level. On the other hand, it requires the user to learn the C API of R. Especially, when an R interface is built on top of existing C/C++ code this constitutes an additional effort. Since R has no Fortran API, the modern interfaces to C/C++ code are not suitable to embed Fortran code into R. Second, the *foreign function interface* provides the R functions `.C()` and `.Fortran()`. This interface allows the compiled code to read and modify atomic R vectors, which are exposed as the corresponding C/C++ and Fortran types, respectively. Thus, no additional API is required, making it favorable for embedding C/C++ and Fortran code that is not specifically designed for R.

On top of these interfaces provided by R, R packages exist that simplify the integration of compiled code into R. One such R package is **inline** (Sklyar et al., 2016), which allows the user to dynamically define R functions and S4 methods with inlined compiled code. Other examples are **Rcpp** (Eddelbuettel et al., 2016a; Eddelbuettel and François, 2011; Eddelbuettel, 2013) and its extensions **RcppArmadillo** (Eddelbuettel et al., 2016b; Eddelbuettel and Sanderson, 2014), **RcppEigen** (Bates et al., 2016; Bates and Eddelbuettel, 2013), **RcppParallel** (Allaire et al., 2016), and **Rcpp11** (François and Ushey, 2014), which greatly simplify the extension of R with C++ code. Similar to the modern interfaces to C/C++ code, the **Rcpp** package family is designed to extend R with compiled code that is specifically written for that purpose.

Building R packages is a way to share compiled code across different platforms. (See, e. g. , Plummer, 2011 for comments on including portable C++ code in R packages.) As of 09-02-2016, 2′303 of the 9′079 R packages on CRAN (`http://www.cran.r-project.org/`) include compiled C/C++ and/or Fortran code using both the foreign function interface and the modern interfaces to C/C++ code with a similar frequency. Figures 1 gives an overview of the number of packages using `.C()`, `.Fortran()`, `.Call()`, and `.External()`.

In the remainder of this article, we focus on the intention to embed compiled code into R without using its C API. An example of an R package using that type of interface is the SPArse Matrix package **spam** (Furrer, 2016; Furrer and Sain, 2010; Gerber and Furrer, 2015), which is built around the Fortran library SPARSKIT (Saad, 1994). Here, the R function `.Fortran()` from the foreign function interface seems to be suitable. Conversely, using the modern interfaces to C/C++ code is also possible but requires adding an additional layer of C code to enable communication between R and the compiled Fortran code. However, using `.Fortran()` is also not satisfying, since it lacks flexibility and

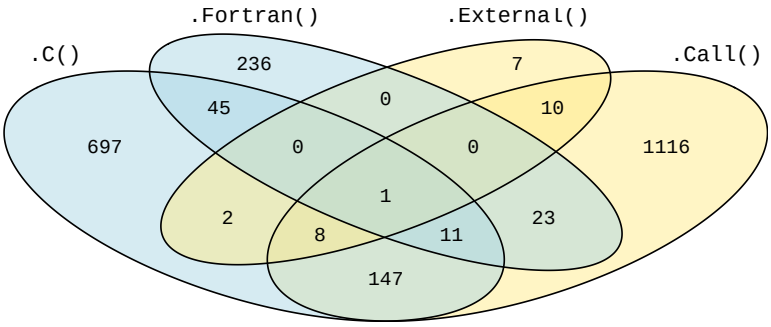


Figure 1: Number of R packages on CRAN using the indicated R functions to interface C/C++ or Fortran code (as of 2016-09-02). Note that R packages linking to **Rcpp** use `.Call()`.

performance, as also stated in its help page: "These functions [`.C()` and `.Fortran()`] can be used to make calls to compiled C and Fortran 77 code. Later interfaces are '.Call' and '.External' which are more flexible and have better performance." Two of the missing features of the foreign function interface are:

- support of long vectors,
- a mechanism to avoid unnecessary copies of R vectors.

The latter is the reason for the lower performance of the foreign function interface compared to the modern interfaces to C/C++ code. Since the foreign function interface does not allow R vectors to be passed to compiled code by reference (without copying), it is especially impractical for big data application. The missing features of the foreign function interface motivated the development of the R package **dotCall64** (Mösinger, Gerber, and Furrer, 2016), which is presented in this article.

Limitations of the foreign function interface

To set the scene for **dotCall64**, we first discuss some limitations of the foreign function interface and give insights into the R implementation of long vectors.

Long vectors

The foreign function interface does not support long vectors; see `help("long vector")`. To understand why extending it to support long vectors is a non-trivial task, we give more details on the long vector implementation of R. In R, vectors are one of the most basic object types underlying more complex objects, such as matrices and arrays. They can be thought of as strings of elements that can be indexed according to their relative positions. Prior to version 3.0.0, the length of vectors was limited to $2^{31} - 1$ elements and indexing thereof was exclusively based on R vectors of type integer. More precisely, the latter are signed 32-bit integer vectors having a value range of $[-2^{31} + 1, 2^{31} - 1]$. Starting from the release of version 3.0.0 in early 2013, support for so-called long vectors was supplied. That is, atomic (`raw`, `logical`, `integer`, `numeric`, `complex`, and `character`) vectors, `lists`, and `expressions` can now have up to 2^{52} elements. The introduction of long vectors was done with minimal changes in R and especially, without changing or adding a 64-bit integer data type. Vectors of lengths less than $2^{31} - 1$ remain unchanged and addressing elements thereof still uses R vectors of type `integer`. In contrast, long vectors use `numeric` vectors of type `doubles` to address elements, which are integer precise up to 2^{52}. This implied changes in some R functions, such as `length()`, which returns an `integer` or a `double` type depending on whether the input vector is a long vector.

```
> typeof(length(integer(1)))
[1] "integer"
> typeof(length(integer(2^31)))
[1] "double"
```

Note that `as.numeric()` returns a `double` type and `as.integer()` returns an `integer` type, though both `integer` and `double` type are of class `"numeric"`, see the "Note on names" section in the R help page `help("is.double")`.

While the R implementation of long vectors favors backwards compatibility, care is needed when manipulating those with compiled code. We distinguish between passing long vectors and indexing long vectors: The former requires passing vectors of more than $2^{31} - 1$ elements to complied code and

is trivial. The latter is challenging, since the indexing R vector is of type `double`, whereas the compiled code would naturally expect a 64-bit integer type. To overcome this discrepancy, one needs to cast the indexing vector from a `double` to a 64-bit integer type before calling the compiled code and back-cast it afterwards.

Technical note: This section gives technical insights into the underlying C implementation of long vectors in R and may be skipped without loss of the general idea. We refer to the source code of R version 3.3.1 in several places and show relevant parts thereof in the appendix. Information on the current and future directions of long vectors and 64-bit types in R can be found in "R Internals" (R Core Team, 2016b, Section 12).

In R, vectors are made out of a header of type `VECSEXP` that is followed by the actual data (Listing 1, line 272). The header contains a field `length` of type `R_len_t`, which is defined as signed `int32_t` (a 32-bit integer). Thus, that `length` field cannot capture the length of a long vector. Instead, it is set to `-1` whenever the length of the vector is larger than $2^{31} - 1$, and an additional header of type `R_long_vec_hdr_t` is prefixed. The prefixed header has a field `length` of type `R_xlen_t`, which is defined as `ptrdiff_t` type (Listing 1, line 75) being "[...] the signed `integer` type of the result of subtracting two pointers. This will probably be one of the standard signed `integer` types (short int, int or long int), but might be a nonstandard type that exists only for this purpose" (GNU C Library, 2016, Appendix A.4).

This implementation has the advantage that the existing code does not need to be changed and still works with vectors having less than 2^{31} elements. Hence, the C code of R can be changed successively to support long vectors throughout several R versions, as opposed to changing the entire C code in one step. To make C code compatible with long vectors, adaptations are needed. For example, the widely used C function `R_len_t length(SEXP s)` (Listing 2, line 124) returns the length of a `SEXP` (S expression) as a `R_len_t`. Thus, all instances of that function have to be replaced with calls to the 64-bit counterpart (i. e. , the function `R_xlen_t xlength(SEXP s)` given in line 159 of Listing 2).

Copying arguments

The foreign function interface exposes pointers to R vectors to compiled code. In order to avoid any corruption of R vectors, they are copied and the compiled code receives pointers to copies of the R vectors. One exception is when the R vector has the named status 0 (i. e. , the object is not bound to any symbol); see "Writing R Extensions" (R Core Team, 2016a, Section 5.9.10). This is the case when the passed R vector is an evaluated constructor (e. g. , `integer(1)`). This is often used when the only purpose of the R vector is to capture results from the compiled code.

Another situation in which there is no need for copying R vectors is when the compiled code only reads an R vector without modifying it. However, the foreign function interface does not allow the user to avoid copying of R vectors (with named status 1 or 2), which leads to a significant computational overhead, especially for large vectors. Note that prior to R version 3.2.0, the copying of R vectors could be avoided by setting the argument `DUP` of `.C()` and `.Fortran()` to `FALSE`. In later R versions, this argument is depreciated and users are referred to the modern interfaces to C/C++ code as a more flexible interface; see `help(".C")` and "R NEWS" (R Core Team, 2016c).

The R package dotCall64

The limitations of the foreign function interface discussed above have motivated the development of the R package **dotCall64**. Its main function is `.C64()`, which can be used to interface compiled code. In contrast to `.C()` and `.Fortran()`, it supports long vectors and 64-bit integer arguments of complied compiled functions/subroutines and provides a mechanism to control duplication of function arguments. Emphasis was put on providing a trustworthy implementation featuring structured R and C source code, documentation, examples, unit tests implemented with **testthat** (Wickham, 2011), and R scripts containing the later presented performance measurements.

Usage of the R function `.C64()`

The function `.C64()` can be used as an enhanced replacement of the foreign function interface and is equally easy to use; see also the documentation in the reference manual (Mösinger, Gerber, and Furrer, 2016). Its syntax resembles that of the function `.C()`, and both functions have common arguments as shown in Table 1.

.C()		.C64()	
arguments	defaults	arguments	defaults
.NAME		.NAME	
		SIGNATURE	
'...'		'...'	
		INTENT	NULL
NAOK	FALSE	NAOK	FALSE
*DUP	TRUE		
PACKAGE		PACKAGE	""
*ENCODING			
		VERBOSE	getOption("dotCall64.verbose")

Table 1: Arguments and default values of the R function `.C()` from the foreign function interface and `.C64()` from **dotCall64**. The depreciated arguments of `.C()` are marked with "*".

SIGNATURE	C/C++ type	Fortran type	R type	cast
"double"	double	double precision	double	no
"int"	int	integer (kind = 4)	integer	no
"int64"	int64_t	integer (kind = 8)	double	yes

Table 2: Supported `SIGNATURE` arguments of `.C64()` and the corresponding C/C++, Fortran, and R data types. The column "cast" indicates whether casting is necessary.

The required arguments of `.C64()` are:

`.NAME` The name of the compiled C/C++ function or Fortran subroutine.

`...` Up to 65 R vectors to be accessed by the compiled code.

`SIGNATURE` A character vector of the same length as the number of arguments of the compiled function/subroutine. Each string specifies the signature of one such argument. Accepted signatures are `"integer"`, `"double"`, and `"int64"`. The R, C/C++, and Fortran types corresponding to these specifications are given in Table 2.

With that, the following call to the compiled C function `void get_c(double input, int index, double output)` using `.C()` can be replaced by its `.C64()` counterpart. Therefore, for example,

```
> .C("get_c", input = as.double(1:10), index = as.integer(9), output = double(1))
```

becomes

```
> .C64("get_c", SIGNATURE = c("double", "integer", "double"),
+      input = 1:10, index = 9, output = 0)
```

While more detailed code examples are given later, this is enough to highlight some features of `.C64()`. First, `.C64()` does require the additional argument `SIGNATURE` specifying the argument types of the compiled function/subroutine. In return, it coerces the provided R vectors to the specified signatures making the `as.double()` and `as.integer()` statements unnecessary. Second, all provided arguments can be long vectors. Third, if one of the arguments of the compiled function is a 64-bit integer (`int64_t` in the case of C/C++ functions, and `integer (kind = 8)` types for Fortran subroutines), it is enough to set the corresponding `SIGNATURE` argument to `"int64"` to successfully evaluate the function. That is, `.C64()` does the necessary `double` to 64-bit integer and 64-bit integer to `double` castings before and after evaluating the compiled code, respectively.

Additional arguments of `.C64()` are the following:

`INTENT` A character vector of the same length as the number of arguments of the compiled function/subroutine. Each string specifies the intent of one such argument. Accepted intents are `"rw"` (read and write), `"r"` (read), and `"w"` (write).

`NAOK` A logical flag specifying whether the R vectors passed though '...' are checked for missing and infinite values.

`PACKAGE` A character vector of length one restricting the search path of the compiled function/subroutine to the specified package.

`VERBOSE` If `0` (default), no warnings are printed. If `1` and `2`, then warnings for tuning and debugging purposes are printed.

A complete list of arguments including their default values is also given in Table 1.

The argument INTENT influences the copying of R vectors and can be seen as an enhanced version of the depreciated DUP argument of .C(). By default, all intents are set to "read and write" implying that the compiled code receives pointers to copies of the R vector given to '...'. This behavior is desirable when the compiled function reads the corresponding R vectors and modifies (writes to) them. For arguments of the compiled function/subroutine that are only read and not modified, the intent can be set to "read." With that, the compiled code receives pointers to the corresponding R vectors itself. While this avoids copying, it is absolutely necessary that the compiled code does not alter these vectors, as this corrupts the corresponding R vectors in the current R session. For arguments that are only used to write results into it, the intent "write" is suitable. To obtain the desired performance gain, the corresponding R vectors passed to '...' have to be of class "vector_dc". R objects of that class contain information on the type and length of the vectors. They can be constructed with the R function vector_dc(), taking the same arguments as vector() from the **base** R package. For example, instead of passing the R vector vector(mode = "numeric", length = 8), the following R object should be passed.

```
> vector_dc(mode = "numeric", length = 8)
$mode
[1] "numeric"

$length
[1] 8

attr(,"class")
[1] "vector_dc" "list"
```

Based on this information, .C64() allocates the corresponding vector (initialized with zeros). That vector is then exposed to the compiled function to write into it. Note that specifying the suitable intent may reduce computation time by avoiding unnecessary copying of R vectors and by avoiding unnecessary double to 64-bit integer and 64-bit integer to double castings for SIGNATURE = "int64" type arguments. More details on the other arguments are given in the package manual of dotCall64 (Mösinger, Gerber, and Furrer, 2016).

Implementation of the R function .C64()

The function .C64() uses the function .External() from the modern interfaces to C/C++ code to directly pass all provided arguments to the C function dC64(). After basic checks of the provided arguments, the function proceeds as schematized in Figure 2. Note that the flowchart depicts the procedure for the case in which the compiled function/subroutine has only one argument. Otherwise, dC64() repeats the depicted scheme for all arguments.

One aspect to highlight is the castings of R vectors for SIGNATURE = "int64" arguments. For such arguments, the double to int64_t casting is done for the intents "read and write" and "read"; see the boxes labeled with (a). In that case, duplication is not necessary, as the implemented casting allocates a new vector anyway. The back-casting from int64_t to double is only done for the intents "read and write" and "write"; see the box labeled with (b).

Moreover, an argument of SIGNATURE different from "int64" with intent "read and write" is duplicated in any case; see boxes labeled with (c). If the intent is "read," it is not duplicated, and if the intent is "write," the argument is only duplicated when it has a reference status different from 0. R vectors increase their reference status when they are passed to an R function, and therefore a safe way to allocate a zero initialized vector without copying is to pass an R object of class "vector_dc".

As casting is an expensive operation in terms of computational time, we distribute this task to multiple threads using openMP, if available (Dagum and Menon, 1998; OpenMP architecture review board, 2016). Note that the number of used threads can be controlled with the R function omp_set_num_threads() from the package **OpenMPController** (Guest, 2013). The package **dotCall64** can also be compiled without the openMP feature by removing the flag '$(SHLIB_OPENMP_CFLAGS)' in the 'src/Makevars' file of the source code.

Examples

We showcase the function .C64() from the R package **dotCall64** with an example function implemented in C and Fortran. Besides the calls thereof via .C64(), the C and Fortran function definitions and the commands to compile and load the code are given. A direct comparison with .C() shows the limitations of the foreign function interface and that it is straight forward to overcome these with .C64(). Moreover, the similarities and differences in the syntax become visible. The considered

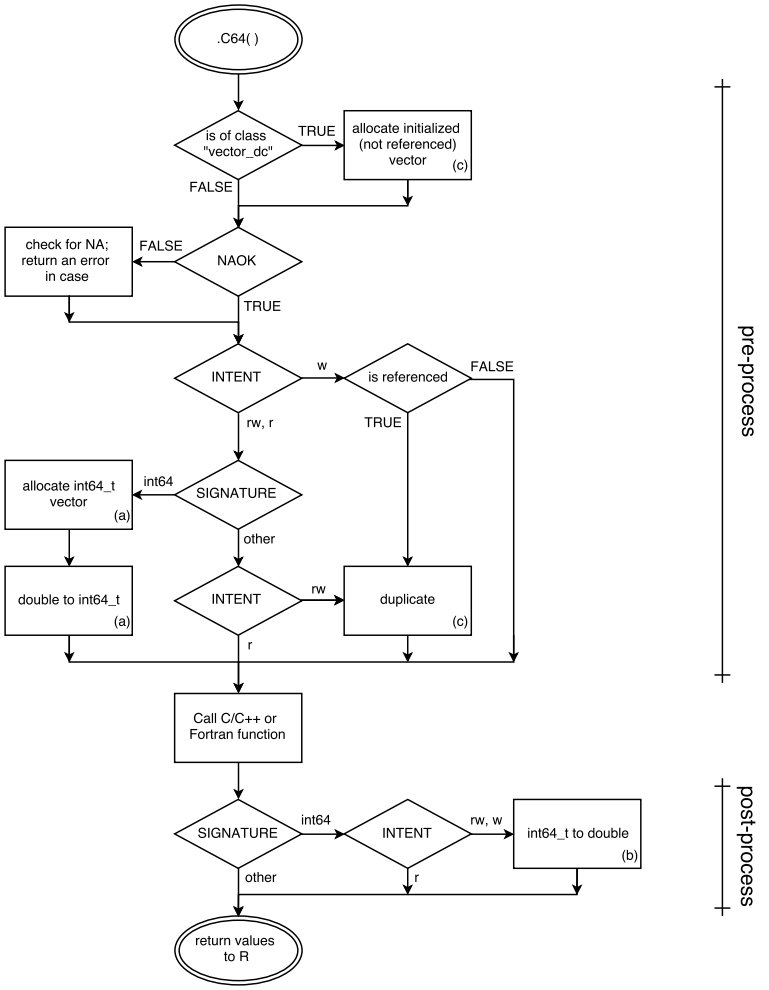


Figure 2: Flowchart of the involved processes when using `.C64()` to call a compiled function/ subroutine with one argument. In the pre-process phase, the provided R vector passed through '`...`' is checked and prepared according to the arguments `NAOK`, `SIGNATURE`, and `INTENT`. Then, the compiled function/subroutine specified with the argument `.NAME` is called. Finally, the vector is back-cast in the post-process phase if necessary.

example function takes the arguments '`input`' (`double`), '`index`' (`integer`), and '`output`' (`double`) and writes the element of '`input`' at the position specified with '`index`' to '`output`'.

Interface C/C++ code

A C implementation of the described example function is given next.

```
void get_c(double *input, int *index, double *output) {
    output[0] = input[index[0] - 1];
}
```

We write the function into '`get_c.c`' and compile it with the command line command '`R CMD SHLIB get_c.c`'. The resulting dynamic shared object ('`get_c.so`' on our Linux platform) must be loaded into R before the compiled function can be called. Note that, in the following R code, the extension of the shared object is replaced with `.Platform$dynlib.ext` to make the code platform independent.

```
> dyn.load(paste0("get_c", .Platform$dynlib.ext))
```

One can use the foreign function interface to call this function. We use the R functions `as.double()` and `as.integer()` to ensure that the types of the passed R vectors match the signature of the C function `get_c()`.

```
> .C("get_c", input = as.double(1:10), index = as.integer(9), output = double(1))$output
[1] 9
```

Next, we try to use the same call with a long vector `x_long` passed to the argument '`input`' of `get_c()`.

```
> x_long <- double(2^31); x_long[9] <- 9; x_long[2^31] <- -1
> .C("get_c",
+    input = as.double(x_long), index = as.integer(9), output = double(1))$output
Error: long vectors (argument 1) are not supported in .Fortran
```

As expected, `.C()` throws an error because it does not support long vectors. The error—and the confusing error message referring to `.Fortran()` instead of `.C()`—can be avoided by replacing `.C()` with `.C64()`. This allows the evaluation of the C function `get_c()` with the long vector `x_long`. Additionally, `.C64()` requires the argument `SIGNATURE` encoding the signatures of the arguments of `get_c()`. This information is used to coerce all provided R vectors to the specified signatures. Thus, it is no longer necessary to reassure that the types of the passed R vectors match the signature of the compiled function.

```
> install.packages("dotCall64")
> library("dotCall64")
> .C64("get_c", SIGNATURE = c("double", "integer", "double"),
+      input = x_long, index = 9, output = double(1))$output
[1] 9
```

In contrast to the call using `.C()`, the ninth element of the long vector `x_long` is returned. However, the argument '`index`' of `get_c()` is of type `int` (a 32-bit integer), and hence, elements at positions beyond $2^{31} - 1$ cannot be extracted. To overcome this, we adapt the definition of the C function `get_c()` and replace the `int` type in the declaration of the argument '`index`' with the `int64_t` type, which is defined in the C header file '`stdint.h`'.

```
#include <stdint.h>
void get64_c(double *input, int64_t *index, double *output) {
    output[0] = input[index[0] - 1];
}
```

We write the function into '`get64_c.c`' and compile it with '`R CMD SHLIB get64_c.c`' to obtain the dynamic shared object ('`get64_c.so`' on our platform). Because of the `int64_t` argument, it is not possible to call this function with `.C()`. On the other hand, `.C64()` can interface this function when the second element of the `SIGNATURE` argument is set to `"int64"`.

```
> dyn.load(paste0("get64_c", .Platform$dynlib.ext))
> .C64("get64_c", SIGNATURE = c("double", "int64", "double"),
+      input = x_long, index = 2^31, output = double(1))$output
[1] -1
```

In the call above, the function `.C64()` casts the argument '`index`' from `double` (the R representation of 64-bit integers) into a `int64_t` type vector before calling `get64_c()`, and back-casts it from `int64_t` to `double` afterwards.

Interface Fortran code

The function `.C64()` can also be used to interface compiled Fortran code. To highlight some Fortran specific features, we translate the C function `get_c()` into the Fortran subroutine `get_f()`.

```
      subroutine get_f(input, index, output)
      double precision :: input(*), output(*)
      integer :: index
      output(1) = input(index)
      end
```

Note that we only use lower case letters in the Fortran function and variable names to avoid unnecessary symbol-name translations. We write the function into the '`get_f.f`' and compile it with '`R CMD SHLIB get_f.f`' to obtain the dynamic shared object ('`get_f.so`' on our platform). In contrast to `.Fortran()`, `.C64()` allows passing pointers to long vectors.

```
> dyn.load(paste0("get_f", .Platform$dynlib.ext))
> .C64("get_f", SIGNATURE = c("double", "integer", "double"),
+      input = x_long, index = 9, output = double(1))$output
[1] 9
```

Again, elements with positions beyond $2^{31} - 1$ cannot be accessed, since the argument '`index`' is of type `integer` and compiled as a 32-bit integer by default. To make `get_f()` compatible with 64-bit

integers, we can either change the declaration of 'index' to 'integer (kind = 8) index' in 'get_f.f' or leave the Fortran code unchanged and set the following compiler flag to compile integers as 64-bit integers.

```
MAKEFLAGS="PKG_FFLAGS=-fdefault-integer-8" R CMD SHLIB get_f.f
```

Note that both the 'kind = 8' declaration and the '-fdefault-integer-8' flag are valid for the GFortran compiler (GNU Fortran compiler, 2014) and may not have the intended effect using other compilers. The resulting dynamic shared object from the command above ('get_f.so' on our platform) can be called from R as follows.

```
> dyn.load(paste0("get_f", .Platform$dynlib.ext))
> .C64("get_f", SIGNATURE = c("double", "int64", "double"),
+      input = x_long, index = 2^31, output = double(1))$output
[1] -1
```

Extend R packages to support long vectors

Extending R packages to support long vectors allows developers to distribute compiled code featuring 64-bit integers with an R user interface. Given the popularity of R, this is a promising approach to make such software available to many users. With the function `.C64()`, the workload of extending an R package to support long vectors is reduced to the following tasks:

- replace the R function to call compiled code with `.C64()`,
- replace the 32-bit integer type declarations in the compiled code with a 64-bit integer declaration.

The latter task implies replacing all `int` type declarations in C/C++ code with `int64_t` type declarations and replacing all `integer` type declarations in Fortran code with 'integer (kind = 8)'. In both cases, the replacements can be automatized (e. g. , with the stream editor GNU sed, 2010). If the considered Fortran code does not explicitly declare the bits of the integers, an alternative approach is to set the compiler flag '-fdefault-integer-8' to compile integers as 64-bit integers using GFortran compilers. This is convenient because the Fortran code does not need to be changed at all in that case.

A more elaborate extension could feature two versions of the compiled code: one with 32-bit integers and the other one with 64-bit integers. Then, the R function can dispatch to either version according to the sizes of the involved vectors. This avoids `double` to 64-bit integer castings when only vectors with less than $2^{31} - 1$ elements are involved. It is convenient to manage two versions of compiled code by putting them into two separate R packages. The first package includes the compiled code with 32-bit integers together with the R code and the documentation. This package can be used independently as long as no long vectors are involved. The second package can be seen as an add-on package and includes only the compiled code with integers declared as 64-bit integers. Thus, loading both packages enables long vector support. This separation into two packages has the advantage that the compiled functions featuring 32-bit integers and their 64-bit counterparts can have the same name. The desired function is then specified by setting the appropriate `PACKAGE` argument of `.C64()`.

As a proof of concept, we extended the sparse matrix algebra R package **spam** to handle sparse matrices with more the $2^{31} - 1$ non-zero elements. From the user perspective, the syntax to manipulate such matrices remains the same. In fact, **spam** users may not even notice the extension. In the case, in which the number of non-zero entries of a matrix exceeds $2^{31} - 1$ and the add-on package **spam64** is loaded, **spam** automatically dispatches to the compiled code with 64-bit integers. The new capabilities of **spam** and **spam64** were illustrated with a parametric model of a non-stationary spatial covariance matrix fitted to satellite data. More information on **spam64** and the data example is given by Gerber, Mösinger, and Furrer (2016).

Performance

There are different settings in which the elapsed time to interface compiled code is relevant. One of those is when the compiled code is interfaced often and takes only a short time to evaluate. Here, the overhead of the interface becomes relevant, which is in the order of a few microseconds for `.C64()`. Another such setting is when large and possibly long vectors are passed through `.C64()`. In that case, the overhead is negligible, as other services of the interface and the execution of the compiled code take up several orders of magnitude more time. When `.C64()` is used to interface 64-bit integer arguments of the compiled code, the largest share of the elapsed time is caused by the `double` to 64-bit integer and 64-bit integer to `double` castings. Since castings are implemented with openMP, the elapsed time thereof also depends on the number of used threads. Besides that, copying objects and checking them for missing/infinite values are also time-consuming operations.

Another performance aspect is peak memory usage. Using the default arguments of `.C64()`, its peak memory usage is about twice the size of the R vectors passed through '...', and hence, is similar to `.C()`. An exception where the peak memory usage is reduced is indicated below.

Performance relevant arguments of `.C64()`

Further, `.C64()` provides arguments to optimize calls to compiled code, one of which is the argument `INTENT`, which is set to "read and write" by default. Since many compiled functions/subroutines only read or write to certain arguments, it is safe to avoid copying in some cases. For example, the C function `get64_c()`, as defined above, only reads the arguments '`input`' and '`index`' and only writes to the argument '`output`'. Thus, we can set the `INTENT` argument of `.C64()` to `c("r","r","w")` and pass the argument with intent "write" as objects of class `"vector_dc"` to reduce the copying of R vectors to a minimum. Another significant performance gain is obtained by setting the argument `NAOK` to `TRUE`. This avoids checking the R vectors passed through '`...`' for `NA`, `NaN`, and `Inf` values. Small-scale performance gains can be achieved by setting the `PACKAGE` argument, which reduces the time to find the compiled code, and by setting `VERBOSE = 0`, which avoids the execution of '`getOptions("dotCall64.verbose")`'. Similar speed considerations that are partially applicable to `.C64()` are given in "Writing R Extensions" (R Core Team, 2016a, Section 5.4.1). An optimized version of the call to the C function `get64_c()`, taking the discussed performance considerations into account, is given next.

```
> .C64("get64_c", SIGNATURE = c("double", "int64", "double"),
+      input = x_long, index = 2^31, output = numeric_dc(1),
+      INTENT = c("r", "r", "w"), NAOK = TRUE, PACKAGE = "dotCall64", VERBOSE = 0)
```

Timing measurements

In the following, we present detailed timing measurements and benchmark `.C64()` against `.C()`, where possible. We consider the following C function contained in the R package **dotCall64**.

```
void BENCHMARK(void *a) { }
```

This function takes one pointer '`a`' to a variable of an unspecified data type and does no operations with it. Thus, the elapsed time to call `BENCHMARK()` from R is dominated by the performance of the used interface. We measure the time to call this function with different `NAOK` and `INTENT` settings of `.C64()` and benchmark it against `.C()` using **microbenchmark** (Mersmann et al., 2015). To get an estimate of the measurement uncertainty, we repeated the measurements between 100 and 10′000 times and report the median elapsed time as well as the interquartile range (IQR) of the replicates. Naturally, timing measurements are platform dependent. We produced the presented results on Intel Xeon CPU E7-2850 2.00 GHz processors using a 64-bit Linux environment where R was installed with default installation flags. When not indicated differently, the measurements were produced using a single thread.

First, we consider the situation in which a pointer to an R vector of length one is passed to the compiled C function `BENCHMARK()`. The following truncated R code illustrates how the measurements were performed. The complete R scripts implementing all presented performance measurements are available in the '`benchmark`' directory in the source code of **dotCall64**.

```
> library("microbenchmark")
> int <- integer(1)
> microbenchmark(
+   .C("BENCHMARK", a = int, NAOK = FALSE, PACKAGE = "dotCall64"),
+   .C64("BENCHMARK", SIGNATURE = "integer", a = int, INTENT = "rw",
+        NAOK = FALSE, PACKAGE = "dotCall64", VERBOSE = 0),
+   .C64("BENCHMARK", SIGNATURE = "integer", a = int, INTENT = "r",
+        NAOK = FALSE, PACKAGE = "dotCall64", VERBOSE = 0),
...
```

Since the R vector '`int`' is very short, a large part of the elapsed time in this experiment is caused by the overhead of the interfaces. Table 3 presents the resulting timing measurements in microseconds. They indicate that `.C()` is more than two times faster compared to `.C64()`. However, this is not surprising, since `.C64()` is more flexible and therefore has a larger overhead. The arguments `NAOK` and `INTENT` have little influence on the elapsed times. The IQRs of around one microsecond indicate a relatively large variability of the elapsed time, which is typical for short timing measurements.

	NAOK = FALSE			NAOK = TRUE		
	.C	.C64 [rw]	.C64 [r]	.C	.C64 [rw]	.C64 [r]
double	2.43 (0.46)	7.11 (0.37)	6.97 (0.40)	2.40 (0.45)	7.04 (0.35)	6.92 (0.37)
integer	2.39 (0.33)	7.54 (0.85)	7.43 (0.85)	2.39 (0.34)	7.52 (0.84)	7.39 (0.83)
64-bit integer		8.98 (1.14)	8.63 (1.19)		8.91 (1.17)	8.58 (1.17)

Table 3: Elapsed times in microseconds to pass double, integer, and 64-bit integer pointers to vectors of length one from R to C using `.C()` and `.C64()`. The used `INTENT` arguments of `.C64()` are indicated in brackets. Reported are median elapsed times of 10′000 replicates. The corresponding IQRs are indicated in parentheses.

We repeated the same experiment with vectors of length 2^{28}. Now, the elapsed times are dominated by services of the interfaces (i. e. , checking for missing/infinite values, copying, and casting). The timings in seconds are presented in Table 3. They indicate that `.C64()` with argument `INTENT = "rw"` and `.C()` showed similar elapsed times. When the intent is set to "read" (`INTENT = "r"`), the elapsed times were reduced and dropped to 0.00 seconds for some configurations. Moreover, not checking for missing/infinite values (`NAOK = TRUE`) decreases the elapsed times across all considered cases. The castings of `SIGNATURE = "int64"` arguments seems to be the most time-consuming task. Note that the IQRs are now smaller relative to the measured timings, because the measured times are larger.

	NAOK = FALSE			NAOK = TRUE		
	.C	.C64 [rw]	.C64 [r]	.C	.C64 [rw]	.C64 [r]
double	2.65 (0.05)	3.16 (0.06)	1.82 (0.02)	1.33 (0.06)	1.33 (0.05)	0.00 (0.00)
integer	1.09 (0.03)	1.09 (0.04)	0.43 (0.01)	0.66 (0.03)	0.66 (0.04)	0.00 (0.00)
64-bit integer		5.21 (0.20)	3.80 (0.06)		3.36 (0.06)	1.97 (0.06)

Table 4: Elapsed times in seconds to pass double, integer, and 64-bit integer pointers to vectors of length 2^{28} from R to C using `.C()` and `.C64()`. The used `INTENT` arguments of `.C64()` are indicated in brackets. Reported are median elapsed times of 100 replicates. The corresponding IQRs are indicated in parentheses.

In a second series of timing measurements, we consider the situation in which a pointer to a vector is passed to the compiled code to write into the vector. We measure the elapsed times of this task as shown in the following truncated R code.

```
> microbenchmark(
+   .C("BENCHMARK", a = integer(2^28), NAOK = TRUE, package = "dotCall64")
+   .C64("BENCHMARK", SIGNATURE = "integer", a = integer(2^28), INTENT = "rw",
+        NAOK = TRUE, package = "dotCall64", VERBOSE = 0)
+   .C64("BENCHMARK", SIGNATURE = "integer", a = integer_dc(2^28), INTENT = "w",
+        NAOK = TRUE, package = "dotCall64", VERBOSE = 0),
...
```

Note the usage of `integer_dc()`, which creates a list containing the length and class of the vector. This information is then used by `.C64()` to create the corresponding vector in C. Table 5 shows the timing measurements for the described setting. As expected using `.C64()` with `INTENT = "w"` reduces the elapsed times compared to `INTENT = "rw"` substantially. Furthermore, `.C()` and `.C64()` with `INTENT = "w"` have similar elapsed times. While `.C()` relies on the reference counting mechanism of R objects to avoid copying ("Writing R Extensions," R Core Team, 2016a), `.C64()` uses the `"vector_dc"` class. The latter has the advantage that one `double` to 64-bit integer casting can be avoided in the `SIGNATURE = "int64"` case.

	.C	.C64 [rw]	.C64 [w]
double	0.87 (0.01)	2.28 (0.13)	0.87 (0.01)
integer	0.44 (0.01)	1.16 (0.06)	0.44 (0.01)
64-bit integer		4.27 (0.03)	2.27 (0.02)

Table 5: Elapsed times in seconds to pass double, integer, and 64-bit integer pointers to vectors of length 2^{28} initialized with zeros from R to C using `.C()` and `.C64()`. The used `INTENT` arguments of `.C64()` are indicated in brackets. Reported are median elapsed times of 100 replicates. The corresponding IQRs are indicated in parentheses.

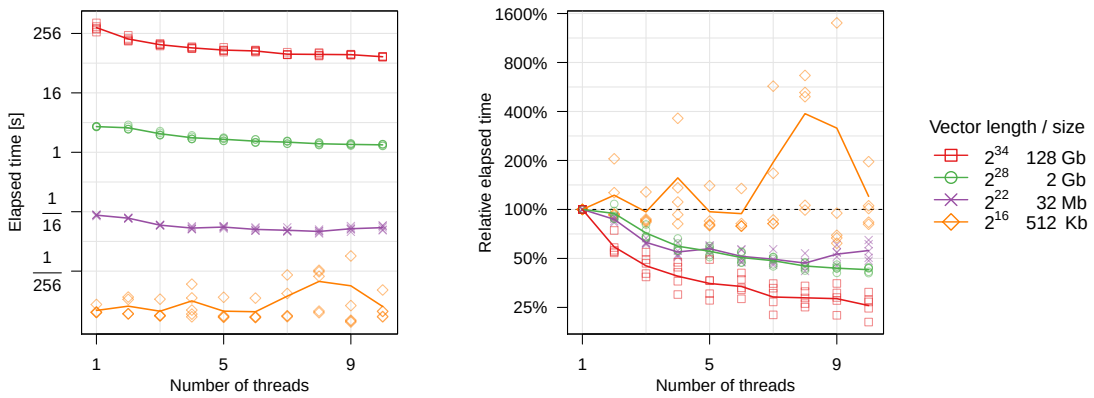


Figure 3: Timings measurements to illustrate the effect of using `.C64()` with enabled multithreading (openMP). Colors and symbols indicate the length/size of the evaluated vectors. Five replicates of each measured configuration are shown with symbols, and the mean values thereof are connected with a line. Left panel: The elapsed time in seconds (y-axis) is plotted against the number of used threads (x-axis). Right panel: The decrease/increase in elapsed time relative to using one thread (y-axis) is plotted against the number of threads (x-axis).

The function `.C64()` features an openMP implementation of the `double` to 64-bit integer and 64-bit integer to `double` castings of `SIGNATURE ="int64"` arguments. Hence, the computational workload of the castings can be distributed to several threads running in parallel. To quantify the performance gain related to using openMP, we control the number of used threads to be between 1 and 10 with the R package **OpenMPController** and measure the elapsed times of the following call.

```
> .C64("BENCHMARK", SIGNATURE = "int64", a = a, INTENT = "rw", NAOK = TRUE,
+      PACKAGE = "dotCall64", VERBOSE = 0)
```

We let 'a' be `double` vectors of length 2^{16}, 2^{22}, 2^{28}, and 2^{34} and performed five replicated timing measurements for each configuration. The results are summarized in Figure 3. The reduction in computation time due to using multiple threads is greatest for the vectors of length 2^{34}, where using 10 threads reduced the elapsed times by about 70%. Conversely, for the vector of length 2^{16} no reduction was observed.

Summary

This paper presents the R package **dotCall64**, which provides an alternative to `.C()` and `.Fortran()` from the foreign function interface. In the first section, we introduce R's interfaces to embed compiled C/C++ and Fortran code. We argue that, in some situations, a `.C()` type interface is more convenient compared to using the C API of R in conjunction with the modern interfaces to C/C++ code. In section two, we motivate the development of **dotCall64** with a discussion of missing features of the foreign function interface and an overview of the R implementation of long vectors. Then, we present the usage and the implementation of the `.C64()` function from the R package **dotCall64**. This is followed by examples demonstrating the capabilities of the new interface—also in comparison with the foreign function interface. Furthermore, we discuss strategies to extend entire R packages with compiled code supporting long vectors. In the last section, we present performance measurements of the `.C64()` interface and benchmark it against `.C()`. This highlights the speed gains achieved by avoiding unnecessary copies of R vectors and by using openMP for casting R vectors. In conclusion, the interface provided by the R package **dotCall64** is an up-to-date version of the foreign function interface including tools to conveniently embed compiled code manipulating long vectors.

Acknowledgments

We thank Rafael Ostertag for contributions to Figure 1. We acknowledge the support of the University of Zurich Research Priority Program (URPP) on "Global Change and Biodiversity."

Bibliography

J. Allaire, R. François, K. Ushey, G. Vandenbrouck, M. Geelnard, and Intel. *RcppParallel: Parallel programming tools for 'Rcpp'*, 2016. URL https://CRAN.R-project.org/package=RcppParallel. R package version 4.3.20. [p1]

D. Bates and D. Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. ISSN 1548-7660. doi: 10.18637/jss.v052.i05. URL https://www.jstatsoft.org/index.php/jss/article/view/v052i05. [p1]

D. Bates, D. Eddelbuettel, R. François, Y. Qiu, and the authors of Eigen for the included version of Eigen. *RcppEigen: 'Rcpp' integration for the 'Eigen' templated linear algebra library*, 2016. URL https://CRAN.R-project.org/package=RcppEigen. R package version 0.3.2.9.0. [p1]

J. M. Chambers. *Software for Data Analysis: Programming with R*. Springer, 2008. ISBN 978-0-387-75936-4. URL http://statweb.stanford.edu/~jmc4/Rbook/. [p1]

L. Dagum and R. Menon. OpenMP: An industry-standard API for shared-memory programming. *IEEE Computing in Science & Engineering*, 5(1):46–55, 1998. ISSN 1070-9924. doi: 10.1109/99.660313. [p5]

D. Eddelbuettel. *Seamless R and C++ integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7. URL http://www.rcpp.org/book/. [p1]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. ISSN 1548-7660. doi: 10.18637/jss.v040.i08. URL https://www.jstatsoft.org/index.php/jss/article/view/v040i08. [p1]

D. Eddelbuettel and C. Sanderson. RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, 2014. doi: 10.1016/j.csda.2013.02.005. [p1]

D. Eddelbuettel, R. François, J. Allaire, K. Ushey, Q. Kou, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ integration*, 2016a. URL https://CRAN.R-project.org/package=Rcpp. R package version 0.12.6. [p1]

D. Eddelbuettel, R. François, and D. Bates. *RcppArmadillo: 'Rcpp' integration for the 'Armadillo' templated linear algebra library*, 2016b. URL https://CRAN.R-project.org/package=RcppArmadillo. R package version 0.7.200.2.0. [p1]

R. L. Eubank and A. Kupresanin. *Statistical Computing in C++ and R*. Chapman & Hall/CRC, 2011. ISBN 9781420066500. [p1]

R. François and K. Ushey. *Rcpp11: R and C++11*, 2014. URL https://CRAN.R-project.org/package=Rcpp11. R package version 3.1.2.0. [p1]

R. Furrer. *spam: SPArse Matrix*, 2016. URL https://CRAN.R-project.org/package=spam. R package version 1.4-0. [p1]

R. Furrer and S. R. Sain. spam: A sparse matrix R package with emphasis on MCMC methods for Gaussian Markov random fields. *Journal of Statistical Software*, 36(10):1–25, 2010. ISSN 1548-7660. doi: 10.18637/jss.v036.i10. URL https://www.jstatsoft.org/index.php/jss/article/view/v036i10. [p1]

F. Gerber and R. Furrer. Pitfalls in the implementation of Bayesian hierarchical modeling of areal count data: An illustration using BYM and Leroux models. *Journal of Statistical Software*, 63(1):1–32, 2015. ISSN 1548-7660. doi: 10.18637/jss.v063.c01. URL https://www.jstatsoft.org/index.php/jss/article/view/v063c01. [p1]

F. Gerber, K. Mösinger, and R. Furrer. Extending R packages to support 64-bit compiled code: An illustration with spam64 and GIMMS $NDVI_{3g}$ data. *Computers & Geosciences*, 2016. submitted. [p8]

GNU C Library. Reference manual, 2016. URL http://www.gnu.org/software/libc/manual/. [p3]

GNU Fortran compiler. Reference manual for GCC version 4.9.2, 2014. URL https://gcc.gnu.org/onlinedocs/gcc-4.9.2/gfortran/. [p8]

GNU sed. Reference manual, 2010. URL https://www.gnu.org/software/sed/manual/. [p8]

S. Guest. *OpenMPController: Control number of OpenMP threads dynamically*, 2013. URL http://CRAN.R-project.org/package=OpenMPController. R package version 0.1-2. [p5]

O. Mersmann, C. Beleites, R. Hurling, and A. Friedman. *microbenchmark: Accurate timing functions*, 2015. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-2.1. [p9]

K. Mösinger, F. Gerber, and R. Furrer. *dotCall64: Foreign function interface with long vector support*, 2016. URL https://CRAN.R-project.org/package=dotCall64. R package version 0.9-04. [p2, 3, 5]

OpenMP architecture review board. OpenMP application program interface, version 4.5, 2016. URL http://www.openmp.org. [p5]

M. Plummer. Portable C++ for R packages. *The R Journal*, 3(2):60–63, 2011. URL http://journal.r-project.org/archive/2011-2/RJournal_2011-2_Plummer.pdf. [p1]

R Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2016a. URL http://cran.r-project.org/doc/manuals/R-exts.html. R version 3.3.1. [p1, 3, 9, 10]

R Core Team. *R Internals*. R Foundation for Statistical Computing, Vienna, Austria, 2016b. URL https://cran.r-project.org/doc/manuals/r-devel/R-ints.html. R version 3.3.1. [p3]

R Core Team. *R News*. R Foundation for Statistical Computing, Vienna, Austria, 2016c. URL https://cran.r-project.org/doc/manuals/r-release/NEWS.html. [p3]

Y. Saad. Parskit: A basic tool kit for sparse matrix computations, 1994. URL http://www-users.cs.umn.edu/~saad/software/SPARSKIT/index.html. [p1]

O. Sklyar, D. Murdoch, M. Smith, D. Eddelbuettel, and R. François. *inline: Inline C, C++, Fortran function calls from R*, 2016. URL http://CRAN.R-project.org/package=inline. R package version 0.3.14. [p1]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p3]

H. Wickham. *Advanced R*. Chapman & Hall/CRC, 2014. ISBN 9781466586963. [p1]

Florian Gerber
Department of Mathematics
University of Zurich
8057 Zurich
Switzerland
florian.gerber@math.uzh.ch

Kaspar Mösinger
Department of Mathematics
University of Zurich
8057 Zurich
Switzerland
kaspar.moesinger@gmail.com

Reinhard Furrer
Department of Mathematics and
Department of Computational Science
University of Zurich
8057 Zurich
Switzerland
reinhard.furrer@math.uzh.ch

Appendix: R source code

In the following, we show parts of the C source code of R version 3.3.1 to support the understanding of the long vector implementation. More precisely, the lines 26–377 from the file 'Rinternals.h' and the lines 124–191 from the file 'Rinlinedfuns.h' are shown in Listing 1 and Listing 2, respectively. The indicated line numbers in the code refer to the actual line numbers of the corresponding file.

Listing 1: 'R-3.3.1/src/include/Rinternals.h'

```
#ifndef R_INTERNALS_H_
#define R_INTERNALS_H_

// Support for NO_C_HEADERS added in R 3.3.0
#ifdef __cplusplus
# ifndef NO_C_HEADERS
#  include <cstdio>
#  ifdef __SUNPRO_CC
using std::FILE;
#  endif
#  include <climits>
#  include <cstddef>
# endif
extern "C" {
#else
# ifndef NO_C_HEADERS
#  include <stdio.h>
#  include <limits.h> /* for INT_MAX */
#  include <stddef.h> /* for ptrdiff_t */
# endif
#endif

#include <R_ext/Arith.h>
#include <R_ext/Boolean.h>
#include <R_ext/Complex.h>
#include <R_ext/Error.h>  // includes NORET macro
#include <R_ext/Memory.h>
#include <R_ext/Utils.h>
#include <R_ext/Print.h>

#include <R_ext/libextern.h>

typedef unsigned char Rbyte;

/* type for length of (standard, not long) vectors etc */
typedef int R_len_t;
#define R_LEN_T_MAX INT_MAX

/* both config.h and Rconfig.h set SIZEOF_SIZE_T, but Rconfig.h is
   skipped if config.h has already been included. */
#ifndef R_CONFIG_H
# include <Rconfig.h>
#endif

#if ( SIZEOF_SIZE_T > 4 )
# define LONG_VECTOR_SUPPORT
#endif

#ifdef LONG_VECTOR_SUPPORT
    typedef ptrdiff_t R_xlen_t;
    typedef struct { R_xlen_t lv_length, lv_truelength; } R_long_vec_hdr_t;
# define R_XLEN_T_MAX 4503599627370496
# define R_SHORT_LEN_MAX 2147483647
# define R_LONG_VEC_TOKEN -1
#else
    typedef int R_xlen_t;
# define R_XLEN_T_MAX R_LEN_T_MAX
#endif

#ifndef TESTING_WRITE_BARRIER
# define INLINE_PROTECT
#endif

/* Fundamental Data Types:  These are largely Lisp
 * influenced structures, with the exception of LGLSXP,
 * INTSXP, REALSXP, CPLXSXP and STRSXP which are the
 * element types for S-like data objects.
 *
 *      --> TypeTable[] in ../main/util.c for  typeof()
 */

/*  These exact numeric values are seldom used, but they are, e.g., in
 *  ../main/subassign.c, and they are serialized.
*/
#ifndef enum_SEXPTYPE
/* NOT YET using enum:
 *  1)  The SEXPREC struct below has 'SEXPTYPE type : 5'
 *  (making FUNSXP and CLOSXP equivalent in there),
 *  giving (-Wall only ?) warnings all over the place
 * 2) Many switch(type) { case ... } statements need a final `default:'
 *  added in order to avoid warnings like [e.g. 1.170 of ../main/util.c]
 *    "enumeration value `FUNSXP' not handled in switch"
 */
typedef unsigned int SEXPTYPE;

#define NILSXP       0    /* nil = NULL */
#define SYMSXP       1    /* symbols */
#define LISTSXP      2    /* lists of dotted pairs */
#define CLOSXP       3    /* closures */
#define ENVSXP       4    /* environments */
#define PROMSXP      5    /* promises: [un]evaluated closure arguments */
#define LANGSXP      6    /* language constructs (special lists) */
#define SPECIALSXP   7    /* special forms */
#define BUILTINSXP   8    /* builtin non-special forms */
#define CHARSXP      9    /* "scalar" string type (internal only)*/
#define LGLSXP      10    /* logical vectors */
/* 11 and 12 were factors and ordered factors in the 1990s */
#define INTSXP      13    /* integer vectors */
#define REALSXP     14    /* real variables */
#define CPLXSXP     15    /* complex variables */
#define STRSXP      16    /* string vectors */
#define DOTSXP      17    /* dot-dot-dot object */
#define ANYSXP      18    /* make "any" args work.
           Used in specifying types for symbol
           registration to mean anything is okay  */
#define VECSXP      19    /* generic vectors */
#define EXPRSXP     20    /* expressions vectors */
#define BCODESXP    21    /* byte code */
#define EXTPTRSXP   22    /* external pointer */
#define WEAKREFSXP  23    /* weak reference */
#define RAWSXP      24    /* raw bytes */
#define S4SXP       25    /* S4, non-vector */

/* used for detecting PROTECT issues in memory.c */
#define NEWSXP      30    /* fresh node created in new page */
```

```
#define FREESXP     31    /* node released by GC */

#define FUNSXP      99    /* Closure or Builtin or Special */


#else /* NOT YET */
/*------ enum_SEXPTYPE ----- */
typedef enum {
    NILSXP  = 0,  /* nil = NULL */
    SYMSXP  = 1,  /* symbols */
    LISTSXP = 2,  /* lists of dotted pairs */
    CLOSXP  = 3,  /* closures */
    ENVSXP  = 4,  /* environments */
    PROMSXP = 5,  /* promises: [un]evaluated closure arguments */
    LANGSXP = 6,  /* language constructs (special lists) */
    SPECIALSXP  = 7,  /* special forms */
    BUILTINSXP  = 8,  /* builtin non-special forms */
    CHARSXP = 9,  /* "scalar" string type (internal only)*/
    LGLSXP  = 10, /* logical vectors */
    INTSXP  = 13, /* integer vectors */
    REALSXP = 14, /* real variables */
    CPLXSXP = 15, /* complex variables */
    STRSXP  = 16, /* string vectors */
    DOTSXP  = 17, /* dot-dot-dot object */
    ANYSXP  = 18, /* make "any" args work */
    VECSXP  = 19, /* generic vectors */
    EXPRSXP = 20, /* expressions vectors */
    BCODESXP  = 21, /* byte code */
    EXTPTRSXP = 22, /* external pointer */
    WEAKREFSXP  = 23, /* weak reference */
    RAWSXP  = 24, /* raw bytes */
    S4SXP = 25, /* S4 non-vector */

    NEWSXP      = 30,   /* fresh node creaed in new page */
    FREESXP     = 31,   /* node released by GC */

    FUNSXP  = 99  /* Closure or Builtin */
} SEXPTYPE;
#endif

/* These are also used with the write barrier on, in attrib.c and util.c */
#define TYPE_BITS 5
#define MAX_NUM_SEXPTYPE (1<<TYPE_BITS)

// ====================== USE_RINTERNALS section
#ifdef USE_RINTERNALS
/* This is intended for use only within R itself.
 * It defines internal structures that are otherwise only accessible
 * via SEXP, and macros to replace many (but not all) of accessor functions
 * (which are always defined).
 */

/* Flags */


struct sxpinfo_struct {
    SEXPTYPE type      :  TYPE_BITS;/* ==> (FUNSXP == 99) %% 2^5 == 3 == CLOSXP
           * -> warning: `type' is narrower than values
           *              of its type
           * when SEXPTYPE was an enum */
    unsigned int obj   :  1;
    unsigned int named :  2;
    unsigned int gp    : 16;
    unsigned int mark  :  1;
    unsigned int debug :  1;
    unsigned int trace :  1;  /* functions and memory tracing */
    unsigned int spare :  1;  /* currently unused */
    unsigned int gcgen :  1;  /* old generation number */
    unsigned int gccls :  3;  /* node class */
}; /*       Tot: 32 */

struct vecsxp_struct {
    R_len_t length;
    R_len_t truelength;
};

struct primsxp_struct {
    int offset;
};

struct symsxp_struct {
    struct SEXPREC *pname;
    struct SEXPREC *value;
    struct SEXPREC *internal;
};

struct listsxp_struct {
    struct SEXPREC *carval;
    struct SEXPREC *cdrval;
    struct SEXPREC *tagval;
};

struct envsxp_struct {
    struct SEXPREC *frame;
    struct SEXPREC *enclos;
    struct SEXPREC *hashtab;
};

struct closxp_struct {
    struct SEXPREC *formals;
    struct SEXPREC *body;
    struct SEXPREC *env;
};

struct promsxp_struct {
    struct SEXPREC *value;
    struct SEXPREC *expr;
    struct SEXPREC *env;
};

/* Every node must start with a set of sxpinfo flags and an attribute
   field. Under the generational collector these are followed by the
   fields used to maintain the collector's linked list structures. */

/* Define SWITH_TO_REFCNT to use reference counting instead of the
   'NAMED' mechanism. This uses the R-devel binary layout. The two
   'named' field bits are used for the REFCNT, so REFCNTMAX is 3. */
```

```
//#define SWITCH_TO_REFCNT

#if defined(SWITCH_TO_REFCNT) && ! defined(COMPUTE_REFCNT_VALUES)
# define COMPUTE_REFCNT_VALUES
#endif
#define REFCNTMAX (4 - 1)

#define SEXPREC_HEADER \
    struct sxpinfo_struct sxpinfo; \
    struct SEXPREC *attrib; \
    struct SEXPREC *gengc_next_node, *gengc_prev_node

/* The standard node structure consists of a header followed by the
   node data. */
typedef struct SEXPREC {
    SEXPREC_HEADER;
    union {
  struct primsxp_struct primsxp;
  struct symsxp_struct symsxp;
  struct listsxp_struct listsxp;
  struct envsxp_struct envsxp;
  struct closxp_struct closxp;
  struct promsxp_struct promsxp;
    } u;
} SEXPREC, *SEXP;

/* The generational collector uses a reduced version of SEXPREC as a
   header in vector nodes.  The layout MUST be kept consistent with
   the SEXPREC definition.  The standard SEXPREC takes up 7 words on
   most hardware; this reduced version should take up only 6 words.
   In addition to slightly reducing memory use, this can lead to more
   favorable data alignment on 32-bit architectures like the Intel
   Pentium III where odd word alignment of doubles is allowed but much
   less efficient than even word alignment. */
typedef struct VECTOR_SEXPREC {
    SEXPREC_HEADER;
    struct vecsxp_struct vecsxp;
} VECTOR_SEXPREC, *VECSEXP;

typedef union { VECTOR_SEXPREC s; double align; } SEXPREC_ALIGN;

/* General Cons Cell Attributes */
#define ATTRIB(x) ((x)->attrib)
#define OBJECT(x) ((x)->sxpinfo.obj)
#define MARK(x)   ((x)->sxpinfo.mark)
#define TYPEOF(x) ((x)->sxpinfo.type)
#define NAMED(x)  ((x)->sxpinfo.named)
#define RTRACE(x) ((x)->sxpinfo.trace)
#define LEVELS(x) ((x)->sxpinfo.gp)
#define SET_OBJECT(x,v) (((x)->sxpinfo.obj)=(v))
#define SET_TYPEOF(x,v) (((x)->sxpinfo.type)=(v))
#define SET_NAMED(x,v)  (((x)->sxpinfo.named)=(v))
#define SET_RTRACE(x,v) (((x)->sxpinfo.trace)=(v))
#define SETLEVELS(x,v)  (((x)->sxpinfo.gp)=((unsigned short)v))

#if defined(COMPUTE_REFCNT_VALUES)
# define REFCNT(x) ((x)->sxpinfo.named)
# define TRACKREFS(x) (TYPEOF(x) == CLOSXP ? TRUE : ! (x)->sxpinfo.spare)
#else
# define REFCNT(x) 0
# define TRACKREFS(x) FALSE
#endif

#ifdef SWITCH_TO_REFCNT
# undef NAMED
# undef SET_NAMED
# define NAMED(x) REFCNT(x)
# define SET_NAMED(x, v) do {} while (0)
#endif

/* S4 object bit, set by R_do_new_object for all new() calls */
#define S4_OBJECT_MASK ((unsigned short)(1<<4))
#define IS_S4_OBJECT(x) ((x)->sxpinfo.gp & S4_OBJECT_MASK)
#define SET_S4_OBJECT(x) (((x)->sxpinfo.gp) |= S4_OBJECT_MASK)
#define UNSET_S4_OBJECT(x) (((x)->sxpinfo.gp) &= ~S4_OBJECT_MASK)

/* Vector Access Macros */
#ifdef LONG_VECTOR_SUPPORT
    R_len_t NORET R_BadLongVector(SEXP, const char *, int);
# define IS_LONG_VEC(x) (SHORT_VEC_LENGTH(x) == R_LONG_VEC_TOKEN)
# define SHORT_VEC_LENGTH(x) (((VECSEXP) (x))->vecsxp.length)
# define SHORT_VEC_TRUELENGTH(x) (((VECSEXP) (x))->vecsxp.truelength)
# define LONG_VEC_LENGTH(x) ((R_long_vec_hdr_t *) (x))[-1].lv_length
# define LONG_VEC_TRUELENGTH(x) ((R_long_vec_hdr_t *) (x))[-1].lv_truelength
# define XLENGTH(x) (IS_LONG_VEC(x) ? LONG_VEC_LENGTH(x) : SHORT_VEC_LENGTH(x))
# define XTRUELENGTH(x) (IS_LONG_VEC(x) ? LONG_VEC_TRUELENGTH(x) : SHORT_VEC_TRUELENGTH(x))
# define LENGTH(x) (IS_LONG_VEC(x) ? R_BadLongVector(x, __FILE__, __LINE__) : SHORT_VEC_LENGTH(x))
# define TRUELENGTH(x) (IS_LONG_VEC(x) ? R_BadLongVector(x, __FILE__, __LINE__) : SHORT_VEC_TRUELENGTH(x))
# define SET_SHORT_VEC_LENGTH(x,v) (SHORT_VEC_LENGTH(x) = (v))
# define SET_SHORT_VEC_TRUELENGTH(x,v) (SHORT_VEC_TRUELENGTH(x) = (v))
# define SET_LONG_VEC_LENGTH(x,v) (LONG_VEC_LENGTH(x) = (v))
# define SET_LONG_VEC_TRUELENGTH(x,v) (LONG_VEC_TRUELENGTH(x) = (v))
# define SETLENGTH(x,v) do { \
      SEXP sl__x__ = (x); \
      R_xlen_t sl__v__ = (v); \
      if (IS_LONG_VEC(sl__x__)) \
    SET_LONG_VEC_LENGTH(sl__x__,  sl__v__); \
      else SET_SHORT_VEC_LENGTH(sl__x__, (R_len_t) sl__v__); \
  } while (0)
# define SET_TRUELENGTH(x,v) do { \
      SEXP sl__x__ = (x); \
      R_xlen_t sl__v__ = (v); \
      if (IS_LONG_VEC(sl__x__)) \
    SET_LONG_VEC_TRUELENGTH(sl__x__, sl__v__); \
      else SET_SHORT_VEC_TRUELENGTH(sl__x__, (R_len_t) sl__v__); \
  } while (0)
# define IS_SCALAR(x, type) (TYPEOF(x) == (type) && SHORT_VEC_LENGTH(x) == 1)
#else
# define SHORT_VEC_LENGTH(x) (((VECSEXP) (x))->vecsxp.length)
# define LENGTH(x)  (((VECSEXP) (x))->vecsxp.length)
# define TRUELENGTH(x)  (((VECSEXP) (x))->vecsxp.truelength)
# define XLENGTH(x) LENGTH(x)
# define XTRUELENGTH(x) TRUELENGTH(x)
# define SETLENGTH(x,v)   ((((VECSEXP) (x))->vecsxp.length)=(v))
# define SET_TRUELENGTH(x,v)  ((((VECSEXP) (x))->vecsxp.truelength)=(v))
# define SET_SHORT_VEC_LENGTH SETLENGTH
# define SET_SHORT_VEC_TRUELENGTH SET_TRUELENGTH
```

```
# define IS_LONG_VEC(x) 0
# define IS_SCALAR(x, type) (TYPEOF(x) == (type) && LENGTH(x) == 1)
#endif
```

Listing 2: 'R-3.3.1/src/include/Rinlinedfuns.h'

```
INLINE_FUN R_len_t length(SEXP s)
{
    switch (TYPEOF(s)) {
    case NILSXP:
  return 0;
    case LGLSXP:
    case INTSXP:
    case REALSXP:
    case CPLXSXP:
    case STRSXP:
    case CHARSXP:
    case VECSXP:
    case EXPRSXP:
    case RAWSXP:
  return LENGTH(s);
    case LISTSXP:
    case LANGSXP:
    case DOTSXP:
    {
  int i = 0;
  while (s != NULL && s != R_NilValue) {
      i++;
      s = CDR(s);
  }
  return i;
    }
    case ENVSXP:
  return Rf_envlength(s);
    default:
  return 1;
    }
}

R_xlen_t Rf_envxlength(SEXP rho);

INLINE_FUN R_xlen_t xlength(SEXP s)
{
    switch (TYPEOF(s)) {
    case NILSXP:
  return 0;
    case LGLSXP:
    case INTSXP:
    case REALSXP:
    case CPLXSXP:
    case STRSXP:
    case CHARSXP:
    case VECSXP:
    case EXPRSXP:
    case RAWSXP:
  return XLENGTH(s);
    case LISTSXP:
    case LANGSXP:
    case DOTSXP:
    {
  // it is implausible this would be >= 2^31 elements, but allow it
  R_xlen_t i = 0;
  while (s != NULL && s != R_NilValue) {
      i++;
      s = CDR(s);
  }
  return i;
    }
    case ENVSXP:
  return Rf_envxlength(s);
    default:
  return 1;
    }
}
```

PAPER III

Predicting Missing Values in Spatio-Temporal Satellite Data

Florian Gerber, Rogier de Jong, Michael E. Schaepman,

Gabriela Schaepman-Strub & Reinhard Furrer

Paper published on *https://arxiv.org/*

Stable URL: https://arxiv.org/abs/1605.01038

Predicting Missing Values in Spatio-Temporal Satellite Data

Florian Gerber, Rogier de Jong, Michael E. Schaepman, *Senior Member, IEEE,* Gabriela Schaepman-Strub, and Reinhard Furrer

***Abstract*—Time series of remotely sensed optical data often contain data points of low product quality, related to atmospheric contamination or angular configuration for example. After detecting and removing such data points, the resulting data product is sparse and contains so called missing values. This is problematic for applications and signal processing methods that require temporally continuous data sets. To address this sparsity, we present a new gap filling method. We predict each missing value separately based on data points in the spatio-temporal neighborhood around the missing data point. The prediction of the missing values and the estimation of the corresponding prediction uncertainties are based on sorting algorithms and quantile regression. The gap filling method was applied to MODIS NDVI data from Alaska and tested with realistic scenarios featuring between 20% and 50% missing data. Validation against established methods showed that the proposed method has a good performance in terms of the root mean squared prediction error, which was between 0.041 and 0.060 and lower compared to the others methods for all test scenarios (Wilcoxon tests, all p-values $< 10^{-15}$). The method is available in the open-source R package gapfill. We demonstrate its performance using a real data example and show how it can be tailored to specific data sets. The computational workload can be distributed among several computers, rendering the method applicable to large data sets. Due to the flexible software design, users can control and redesign relevant parts with little additional effort. This makes it an interesting tool for gap filling satellite data and for the future development of gap filling methods.**

***Index Terms*—Alaska, gap filling, interpolation, MODIS NDVI, quantile regression, R gapfill, TIMESAT, uncertainty.**

I. Introduction

REMOTE sensing is a technology used to study a wide range of Earth surface processes. In particular, when using observations from optical satellite sensors, image data are often contaminated by clouds. Mean annual cloud cover ranges globally from 0% to almost 100%, depending on geographic location [1]. Cloud cover over vegetated areas may not allow proper parameterization of continuous vegetation change based on observations (see Arctic [2]; Amazon [3]; General [4]). When reconstructing land surface phenology, satellite data are mostly filtered to exclude cloudy data, or interpolated using a variety of methods [5], [6]. However, little attempt has been undertaken to date to use gap filling approaches of spatio-temporal nature of optical satellite data prior to applying retrieval algorithms [7]. Many of the currently existing gap filling methods focus on the reconstruction of the temporal component; for a review on gap free data products see [8]. In this paper, we discuss a new approach to gap filling, and validate its performance using realistic test scenarios derived from existing (gap-prone) spatio-temporal NDVI data.

F. Gerber and R. Furrer was with the Institute of Mathematics, University Zurich, Switzerland e-mail: reinhard.furrer@math.uzh.ch.

R. de Jong and M. E. Schaepman was with the Remote Sensing Laboratories, Department of Geography, University of Zurich, Switzerland.

G. Schaepman-Strub was with the Department of Evolutionary Biology and Environmental Studies, University of Zurich, Switzerland.

R. Furrer was also affiliated with the Institute for Computational Science, University of Zurich, Switzerland.

We acknowledge support of the University of Zurich Research Priority Program (URPP) on Global Change and Biodiversity.

Manuscript received April ??, ????; revised ?MONTH? ?DAY?, ?YEAR?.

A. Missing Values in Satellite Data

One typical example of data with missing values occurs when analyzing the Earth's vegetation using the Moderate Resolution Imaging Spectroradiometer (MODIS). MODIS instruments on Terra and Aqua measure radiation reflected by the Earth surface every one to two days on contrasting orbits (descending/ascending). To retrieve the desired information, the data are pre-processed and transformed into a data product; for example, the MOD13 family of vegetation indices. This pre-processing phase consists of aggregation techniques such as constrained-view-angle maximum-value composites [9] and quality assignments [10]. A resulting data product is MOD13A [11], which comprises several data layers containing values on a regular grid in space and time with a resolution of 500 m and 16 days. However, when only values flagged with "good quality" are considered, the proportion of missing data can be considerable. This may negatively influence the inference of Earth-surface processes or even render certain analysis techniques infeasible.

B. Existing Approaches to Handle Missing Values

The strategies that handle the missing data problem can be divided into two groups. The first group employs statistical data analysis methods that are robust to missing values. The second group predicts the missing values in the data, using an additional processing step before the analysis. This can either be done for a data product containing missing values or as an integral part of the pre-processing. In the following, we give an overview of existing methods that handle satellite data with missing values.

1) Robust Analysis Methods: When describing vegetation using remotely sensed data, the temporal characterization of the process is of great interest. This includes short and long-term trends of the values themselves and of derived quantities such as growing season onset, growing season length, etc.

Many methods exist to study these phenomena by interpreting the satellite data as a collection of spatially-independent time series (one per pixel in the spatial extent). Commonly, the time series of each pixel is smoothed to reduce noise and to fill the missing values. The information of interest is then extracted from this smoothed time series. Hence, the presence of missing values is no longer a problem, as long as the smoothing is not negatively influenced by the missing values. Examples of smoothing methods are the Savitzky-Golay filter method [12], the least-squares fitted asymmetric Gaussian method and the double logistic smooth functions method, which are all implemented in the software TIMESAT [13]. The asymmetric Gaussian method was extended to handle larger temporal gaps by also taking spatial neighboring pixels into account [14]. Other approaches use Fourier analysis [15], [16], locally weighted scatter plot smoothing (LOESS) [17], splines [18], and the CACAO method [19]. The software TiSeG [20] retrieves vegetation time series through interpolation, in combination with pixel-level quality assurance data. These and similar methods were compared by applying them to simulated and observed time series [21]. Other studies compared smoothing methods with a focus on vegetation index time series and the extraction of phenological variables [22], [23], [24]. According to these authors, the accuracies of the different methods depend on the properties of the underlying time series. While criteria to assess the performance of temporal smoothing algorithms in the absence of ground reflectance measurements were proposed [25], we are not aware of established reference data sets to evaluate gap filling algorithms.

Moreover, specialized methods exist to study local trends in vegetation index time series, e.g., the "breaks for additive season and trend" (BFAST) algorithm [26], [27] and the "detecting breakpoints and estimating segments in trend" (DBEST) algorithm [28]. Another method uses a combination of neural networks and Savitzky-Golay filter to obtain near-real time estimation of global LAI, FAPAR, and FCOVER variables from SPOT/VEGETATION satellite data [29]. While all of these methods are based on the temporal correlation of the values, there exist also methods that exploits both the temporal and spatial correlation to study spatial patterns of temporal trends in NDVI data [30].

2) Reconstructing a Complete Data Product: Another strategy to handle missing values is to predict them based on observed data. This is typically performed during an additional step before the product of interest is derived from the data. Several authors discuss the use of geostatistical methods such as kriging and co-kriging for this task [31], [32], [33]. These methods exploit the spatial correlation of the data within images. In the case of co-kriging, information from images observed at different points in time are included as a regression term. The same ideas were used to restore Landsat ETM+ data that exhibit systematically missing strips caused by the Scan Line Corrector (SLC) failure in 2003 [34], [35], [36]. The kriging ideas were extended in the direction of spatio-temporal models via spatio-temporal variograms [37], [38], the combination with generalized additive models [39] and Kalman-filtering [40]. Another family of gap filling methods uses singular spectrum analysis [41], [7] or empirical orthogonal functions [42].

In contrast to the aforementioned methods, the following approaches are not derived from classical time series or geostatistical frameworks. Another proposed method is known as "neighborhood similar pixel interpolate" and uses weighted values from images observed at other points in time [43]. This approach was applied to Landsat ETM+ SLC failure data [44] and combined with a kriging based approach [45]. Based on similar ideas, an algorithmic gap filling method for MODIS EVI data was derived and tested at continental scale [46]. Other methods use linear regression models fitted to a spatio-temporal window around the missing value [47], [48]. These methods rely on the availability of additional land cover data [49], [50], or use the quality flags provided for MODIS data products to predict missing values [51].

C. Outline

We introduce and discuss a new gap filling method. It reconstructs a complete data product, allowing the application of methods that rely on gap free data sets. Similar to the methods presented by [46], [48], [47], we use spatio-temporal subsets around the missing values ("gaps") to predict them. We contribute (1) the formalization of this subset-predict strategy in a generic way and design a software implementation of the method accordingly, and we then (2) present a new instance of such a subset-predict algorithm, providing very accurate fill values for the four investigated test scenarios, and we (3) base the proposed method on a statistical framework, which helps to quantify the uncertainties associated with the predicted values.

We introduce the gap filling method, its validation approaches, and corresponding test data in the following sections. The performance of the proposed method is then assessed with four test scenarios and a realistic MODIS NDVI data example. In addition, we compare the accuracy of the predicted values against those obtained with TIMESAT as well as with the "Gapfill-Python" approach as described in [46].

II. Gap Filling Method

A. Formal Description

The proposed gap filling method assumes that the input data set consists of a four dimensional array having regularly spaced values in time and space. Let $z = z[x, y, s, a]$ denote one value of this input data set A; see Tab. I for an overview of the mathematical objects. The indices $x \in \{1, \ldots, N_x\} = I_x$ and $y \in \{1, \ldots, N_y\} = I_y$ describe the spatial location of z, and $s \in \{1, \ldots, N_s\} = I_s$ is a seasonal index describing the temporal position of z within the year $a \in \{1, \ldots, N_a\} = I_a$. Hence, $A = \{z : x \in I_x, y \in I_y, s \in I_s, a \in I_a\}$, or using the dot notation $A = z[\cdot, \cdot, \cdot, \cdot]$. We define an *image* as a collection of values observed at a given point in time. For example, $z[\cdot, \cdot, s, a]$ is an image of A with $N_x \times N_y$ values observed in season s and year a. The term *pixel* refers one or several values of A at a specific spatial position. For example, the values of the pixel (x, y) denotes the collection of values $z[x, y, \cdot, \cdot]$. If these values are ordered such that they have increasing time stamps, this is a *time-series* with $N_s \times N_a$ values. With that, A consists of $N_s \times N_a$ images having $N_x \times N_y$ values each

TABLE I
LIST OF MATHEMATICAL OBJECTS USED TO DESCRIBE THE GAP FILLING METHOD. TUNING PARAMETERS ARE LABELED WITH "*"

Notation	Explanation
$z = z[x, y, s, a] \in \mathbb{R} \cup \{\text{NA}\}$	One value of the data (observed or missing)
$x \in \{1, \ldots, N_x\} = I_x$	x coordinate of the spatial position of z
$y \in \{1, \ldots, N_y\} = I_y$	y coordinate of the spatial position of z
$s \in \{1, \ldots, N_s\} = I_s$	Temporal position of z within a year
$a \in \{1, \ldots, N_a\} = I_a$	Temporal position of z indicating the year
$\mathcal{A} = z[\cdot, \cdot, \cdot, \cdot]$	Input data array containing observed and missing values
$z_0 = z_0[x_0, y_0, s_0, a_0] \in \mathcal{A}$	One missing value of the data
$\hat{z}_0 \in \mathbb{R}$	Predicted value of z_0
$B = B(z_0, i, \lambda_x, \lambda_y, \lambda_s, \lambda_a) \subseteq \mathcal{A}$	Neighborhood around z_0 for iteration i
$z' = z'[x', y', r'] \in B'$	One value of the projected neighborhood B'
$x' \in \{1, \ldots, N'_x\} = I'_x$	x coordinate of the spatial position of z' within B'
$y' \in \{1, \ldots, N'_y\} = I'_y$	y coordinate of the spatial position of z' within B'
$r' \in \{1, \ldots, N'_r\} = I'_r$	Number of the image within B'
$z'_0 = z'[x'_0, y'_0, r'_0]$	Missing within B' that is predicted
$\lambda_x, \lambda_y, \lambda_s, \lambda_a \in \mathbb{N}$	* Spatio-temporal extent of the initial subset
$\theta_1 \in \mathbb{N}$	* Required number of non-empty images in the subset B
$\theta_2 \in \mathbb{N}$	* Required number of non-missing values in the subset B
$\nu \in \mathbb{N}$	* Minimal number of considered values to estimate the quantile

and the values of each pixels can be seen as a time-series. In total, A has $N_x \times N_y \times N_s \times N_a = N$ values.

We assume that all $z \in \mathcal{A}$ are either observed real values or missing (NA), thus $z \in \mathbb{R} \cup \{\text{NA}\}$. Note that we use the term "missing value" to refer to one single missing data point, whereas "data gap" or "gap" refers to one or more missing values. The determination of missing values is an important and challenging task. Many methods to detect low quality product points exist and different methods may be preferred depending on the considered data product and the goal of the analysis. Therefore, we kept the determination of the missing values as flexible as possible, allowing users to select their favorite method. Often the determination of the missing values is based on continuous or discrete information available on a per-pixel basis. Continuous methods rely on screening mechanisms and are usually thresholded to derive information on the use of a value. At instrument level, quantum efficiency estimators are used to identify "defective pixels", which are consequently flagged as faulty and sometimes interpolated using neighborhood operators [52]. At product level, cloud screening is used to thresholding cloud reflectance from aerosols [53]. A broad variety of other continuous methods exist, but are not discussed here in detail. Discrete information is mostly available in the form of quality flags, related to single values [10]. In our examples, we will use the MODIS quality flags to determine the spatio-temporal behavior of the missing values.

The goal of the gap filling method is to predict all missing values $\{z \in \mathcal{A} : z = \text{NA}\}$. To that end, the following algorithm is repeated for all missing values. Let $z_0 = z_0[x_0, y_0, s_0, a_0] \in \mathcal{A}$ denote one such missing value. To predict its value $\hat{z}_0$, we employ the prediction algorithm schematized in the flow diagram in Fig. 1.a featuring a subset and a prediction component. In brief, the subset component selects a neighborhood $B \subseteq \mathcal{A}$ around (x_0, y_0, s_0, a_0) according to a neighborhood search scheme detailed in Section II-A1. The subsequent prediction component decides whether it is possible to predict the missing value based on B. If so, the missing value is predicted as described in Section II-A2; otherwise, the procedure returns to the subset component and updates the subset parameters to extract a larger subset. This iterative algorithm is repeated until the prediction component decides that it is possible to predict z_0 based on the provided subset B (Fig. 1.a).

1) Subset Component: To be more specific about the search strategy for suitable subsets, let $B = B(z_0, i, \lambda_x, \lambda_y, \lambda_s, \lambda_a)$ be a four dimensional box around the missing value. The following definition shows that the spatial extent of B increases as $i \in \mathbb{N}$ increases.

$$\begin{aligned} &B(z_0, i, \lambda_x, \lambda_y, \lambda_s, \lambda_a) \\ &= \{z[x, y, s, a] \in \mathcal{A} : x_0 - (\lambda_x + i) \leq x \leq x_0 + (\lambda_x + i), \\ &\qquad y_0 - (\lambda_y + i) \leq y \leq y_0 + (\lambda_y + i), \\ &\qquad s_0 - \lambda_s \leq s \leq s_0 + \lambda_s, \\ &\qquad a_0 - \lambda_a \leq a \leq a_0 + \lambda_a\} \end{aligned} \tag{1}$$

In case where z_0 is not close to a boundary of $\mathcal{A}$, the spatial extent of B is $(1 + 2\lambda_x + 2i) \times (1 + 2\lambda_y + 2i)$. i is set to 0 initially and is increased by one whenever B is rejected by the prediction component. The parameters $\lambda_x, \lambda_y, \lambda_s, \lambda_a \in \mathbb{N}$ are a first set of tuning parameters, which we set to $(\lambda_x, \lambda_y, \lambda_s, \lambda_a) = (10, 10, 1, 5)$. They define the initial extent of B (if z_0 is not close to a boundary of $\mathcal{A}$). For example, $\lambda_s = 1$ implies that values having the seasonal indices $s-1$, s, and $s+1$ are included. Tuning parameters have to be set by the users and allow them to tailor the method to a specific data set. The choice of the tuning parameter values can be justified via cross-validation [54], [55], [56].

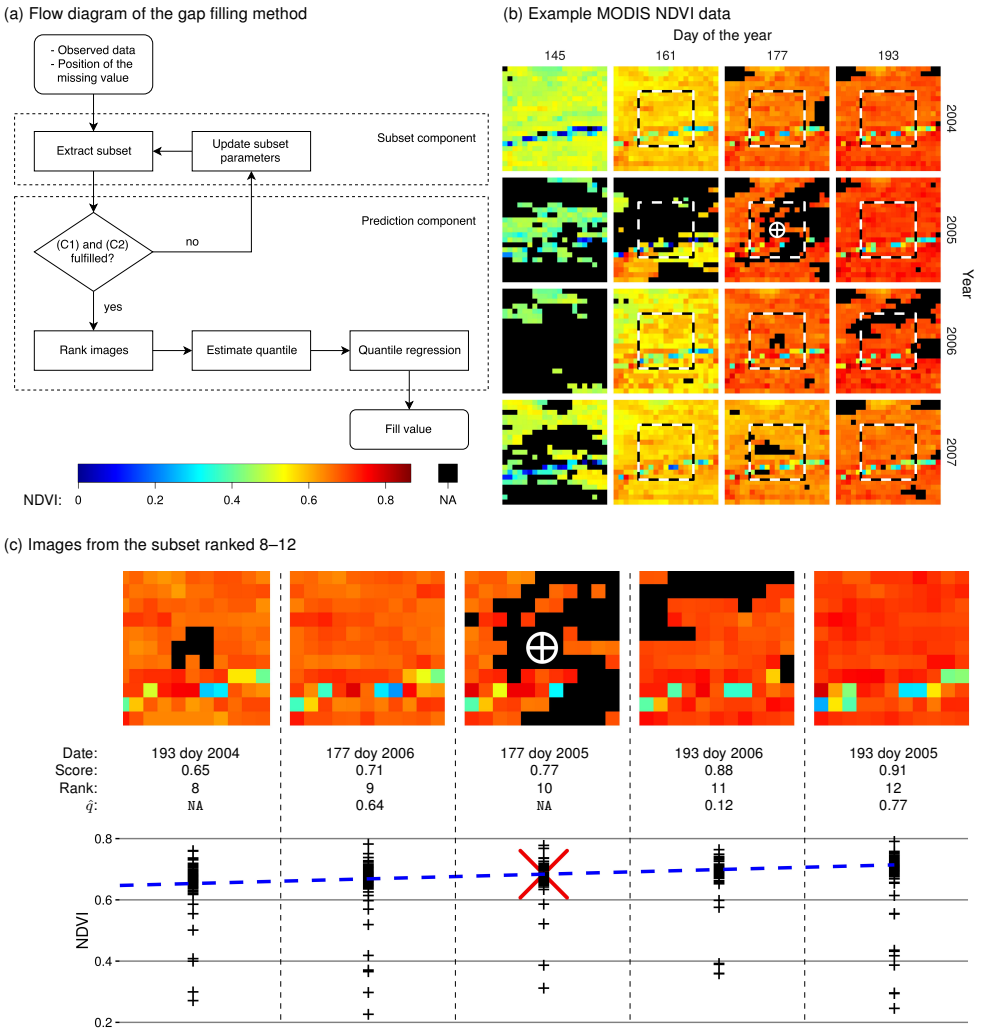


Fig. 1. (a) Flow diagram of the gap filling method showing the involved components to predict one missing value. (b) Example MODIS NDVI data. The depicted images have the spatial extent 21×21. The data contains values observed at 16 points in time (during 4 years, at 4 DOYs). The $1'603$ missing values are depicted in black. To predict the value of DOY 177 of year 2005 marked with a white cross, the subset marked with dashed squares is considered. (c) Insights in the prediction of the missing value from (b) marked with a white cross are given. Depicted are the subsets marked with the dashed lines from (b) ranked 8–12. While the top row depicts the images of that subset, the bottom row shows a scatter plot of the corresponding NDVI values. The dashed blue line is the quantile regression fit and the red cross is the predicted NDVI value.

2) Prediction Component: The first task of the prediction component is to decide whether it is possible to return the predicted value $\hat{z}_0$ based on the provided subset B. In order to return $\hat{z}_0$, both of the following criteria need to be fulfilled:

(C1) B must contain at least θ_1 non-empty images,
(C2) the image in B containing z_0 must have at least θ_2 non-missing values.

θ_1 and θ_2 are again tuning parameters of the method, which we set to 5 and 25, respectively. To explain the actual prediction of a missing value z_0, let B be a neighborhood of z_0, which fulfills the criteria (C1) and (C2). The prediction comprises ranking of the images in B, estimation of the quantile of z_0 relative to the image containing z_0, and quantile regression (Fig. 1.a).

Next, we describe the ranking of the images in B. By construction B is a four dimensional array and can be seen as a collection of images. Some of these images may be empty, i.e., all their values are NA. Let N'_r be the number of non-empty images in B. We proceed with this non-empty images and see them as a collection of N'_r images with no particular temporal ordering. This projected neighborhood is denoted with B'. In other words, B' is constructed by collapsing the two temporal dimensions season and year of B into one dimension and by subsequently removing the empty images. Let $N'_x \times N'_y \times N'_r$ be the extent of B' and let $x' \in \{1, \ldots, N'_x\} = I'_x$, $y' \in \{1, \ldots, N'_y\} = I'_y$, and $r' \in \{1, \ldots, N'_r\} = I'_r$ be indices used to address elements of B'. Then the N'_r images of B' are scored using their non-missing values. Note that scoring of images containing missing

values is not a trivial problem [57]. We assume that the images in B' have a similar but potentially shifted distribution of values and exploit this to construct a scoring algorithm, which is only marginally affected by missing values. The algorithm is based on pixel-wise comparisons of each of the N'_r images in B' with all $N'_r - 1$ other images in B'. Pseudocode 1 describes how the score of one image $k \in I'_r$ is obtained. Note that in the following, mean(X) is the function that returns the mean of all non-missing values of X. By repeating the algorithm for each image of B' all images its are scored. Subsequently, the images of B' are ranked according to their scores.

Pseudocode 1 Score the kth image in B' relative to the other images in B'.

Input: B', k
Output: Score of kth image.

Define M as a $N'_x \times N'_y$ matrix
Define V as a vector of length N'_r, initialized with NA
for $r' \in I'_r \setminus \{k\}$ **do**
  Initialized M with NA
  for $x' \in I'_x, y' \in I'_y$ **do**
    if $z'[x', y', k]$ and $z'[x', y', r']$ are not NA **then**
      if $z'[x', y', k] > z'[x', y', r']$ **then**
        $M[x', y'] \leftarrow 1$
      else
        $M[x', y'] \leftarrow 0$
      end if
    end if
  end for
  $V[r'] \leftarrow$ mean(M)
end for
return mean(V)

To describe the estimation of the α_0-quantile of z_0 relative to the image containing z_0, we again use the dot notation to indicate collections of points, i.e., the entire neighborhood B' is denoted with $z'[\cdot, \cdot, \cdot]$. Further the image in B' with rank r' is denoted with $z'[\cdot, \cdot, r']$ and the missing value of interest is denoted with $z'_0 = z'[x'_0, y'_0, r'_0]$. Moreover, let $\hat{F}_{r'}(x)$ denote the empirical cumulative distribution function estimated from the non-missing values of $z'[\cdot, \cdot, r']$. Let $\hat{\alpha}_{r'} = \hat{F}_{r'}(z[x'_0, y'_0, r'])$ be the estimated quantile of the value $z'[x'_0, y'_0, r']$ relative to the image r'. We estimate the quantile of interest α_0 as the mean of all defined values of $\{\hat{\alpha}_{r'} : r' \in I'_r\}$. Note that some of those values may be undefined because of missing $z'[x'_0, y'_0, r']$ values. We require at least $\nu \in \mathcal{N}$ non-missing values in $\{\hat{\alpha}_{r'} : r' \in I'_r\}$, where ν is a tuning parameter of the method, which we set to 2. If this criterion is not met, all $\hat{F}_{r'}$, $r' \in I'_r$ are also evaluated in a spatial neighborhood of (x'_0, y'_0). A detailed description of the estimation of α_0 is given in Pseudocode 2.

Finally, the prediction of $z_0 = z'_0$ is obtained by fitting a quantile regression to the values of B' having an intercept and the rank r' of the images as linear predictors [58]. The following equation describes the model for the α-quantile $Q(\alpha)$.

$$Q(\alpha \mid r') = \beta_0(\alpha) + \beta_1(\alpha) r' \tag{2}$$

Pseudocode 2 Estimate the α_0-quantile of the missing value relative to the image $z'[\cdot, \cdot, r_0]$.

Input: $B' = z'[\cdot, \cdot, \cdot]$, r_0
Output: $\hat{\alpha}_0$

Define V as a vector of length N'_r, initialized with NA
Set $i \leftarrow 0$, $D \leftarrow z'[x'_0, y'_0, \cdot]$
while the number of non-missing values in $D < \nu$ **do**
  $i \leftarrow i + 1$
  $D \leftarrow \{z'[x', y', \cdot] \in B' : x'_0 - i \leq x' \leq x'_0 + i,$
  $y'_0 - i \leq y' \leq y'_0 + i\}$
end while
Define A as an array with extent $(2i+1) \times (2i+1) \times N'_r$
Initialized A with NA
for $r' \in I'_r$ **do**
  Estimate $\hat{F}_{r'}(\cdot)$ from $z'[\cdot, \cdot, r']$
  for $x' \in \{-i, \ldots, i\}$, $y' \in \{-i, \ldots, i\}$ **do**
    if $z'[x'_0 + x', y'_0 + y', r']$ not is NA **then**
      $A[x'_0 + x', y'_0 + y', r'] \leftarrow \hat{F}_{r'}(z'[x'_0 + x', y'_0 + y', r'])$
    end if
  end for
  $V[r'] \leftarrow$ mean$(A[\cdot, \cdot, r'])$
end for
return mean(V)

In other words, the α-quantile of the values of each images in B' is regressed on the rank of the images r'. As we are only interested in the quantile $\hat{\alpha}_0$ (our guess of the quantile of the missing value z'_0), we can plug it into Equation (2) and estimate the coefficients $\hat{\beta}_0(\hat{\alpha}_0)$ and $\hat{\beta}_1(\hat{\alpha}_0)$ by solving a minimization problem [59]. The predicted value (or fill value) of z'_0 is then $\hat{z}'_0 = \hat{\beta}_0(\hat{\alpha}_0) + \hat{\beta}_1(\hat{\alpha}_0) r'_0$.

B. Illustration with a Test Data Example

For illustration, we consider the MODIS NDVI data (satellite product MOD13A1) shown in Fig. 1.b. Note, however, that the proposed method can easily be applied to a wide range of remotely sensed data. The example data have the spatial extent $N_x \times N_y = 21 \times 21$ and consist of 16 images having $N_s = 4$ seasonal indices (the days 145, 161, 177, and 193 of the year) observed over $N_a = 4$ years (2004–2007). With that, the data array has $N = 7'056$ values in total. We assume the $1'603$ ($\approx 23\%$) values depicted in black to be classified as missing values. The gap filling method predicts the missing values from the observed ones by applying the algorithm illustrated in Fig. 1 to each missing values.

Let us consider the missing value z_0 depicted with a white cross in Fig. 1.b. In the subset component of the algorithm, a subset around that value is selected. For the tuning parameters $(\lambda_x, \lambda_y, \lambda_s, \lambda_a)$=$(5, 5, 1, 5)$ the initial subset $B(i = 0)$ is marked with dashed squares. The subsequent prediction component first decides whether z_0 can be predicted based on B or if the subset needs to be increased. With that mechanism, the spatial extent of the subset is adapted to the local distribution of missing values. This is useful if, for example, a region of the data exhibits a temporally shifted seasonal pattern or a

long-term temporal trend. The selected subset B in Fig. 1.b fulfills the criteria (C1) and (C2) stated in Section II-A2.

Next, the missing value is predicted based on the selected subset. The latter is interpreted as a collection of images, and we rank them according to their values. The rationale behind this is that the ranked series of images imitates the seasonal evolution of the spatial field with an artificially high temporal resolution. The ranking is based on an algorithm that scores the images via pixel-wise comparisons of non-missing values (Pseudocode 1). In Fig. 1.c, the subset of the images ranked 8–12 and their scores are depicted. The missing value of interest is again marked with a white cross. The bottom row of the same panel displays the ordered images as a scatter plot, having the estimated ranks on the x-axis and the observed NDVI values on the y-axis.

The estimation of the α_0-quantile was performed as described in Section II-A2 (Pseudocode 2) with $\nu = 2$ and was estimated to be the 47%-quantile. To finally obtain the prediction of the missing value, linear quantile regression is used. The regression has the observed NDVI values as a response and an intercept as well as the ranks of the image as linear predictors. In Fig. 1.c the fitted regression line is depicted as dashed blue line and the predicted NDVI value is a red cross.

C. Prediction Uncertainties

Uncertainty estimates of the predicted values are essential when using them to derive conclusions in further analyses. Statistical theory provides ways to quantify uncertainty through the estimation of prediction variability and confidence intervals. Possible strategies that can be applied to the proposed gap filling method are based on resampling techniques and cross-validation. While the former is computationally expensive, and therefore difficult to apply to a large data set, the latter was applied to the gap filling method proposed by [46]. However, both approaches are inaccurate, if the underlying assumptions about the data are not met.

Besides that, it is interesting to study the magnitude of the uncertainties introduced by the different steps of the method; the latter are (1) choosing the size of the initial subset around the missing value, (2) ranking the images of the subset (Pseudocode 1), (3) estimating the quantile of the missing values (Pseudocode 2), and (4) estimating the parameters of the quantile regression. We assessed the uncertainties introduced in step (1) by running the gap filling method with all possible initial sizes of the spatial subset. Then a 90% prediction interval for each missing value is obtained by considering the 5% and the 95% quantile of the predicted values. The uncertainty of step (2) is calculated via permutations of the ranked images. More precisely, we approximate all possible permutations of the ranks by changing the rank of the image containing the missing value to all possible positions. The motivation for this approximation is that the permutation of the image containing the missing value has a much larger influence on the predicted value compared to permutations among other images. The predicted values for all such permutations are then calculated and a 90% prediction interval is constructed by considering the 5% and the 95% quantile thereof. For step (3), we derive a 90% prediction interval by quantifying the variability in the estimated image-wise quantiles (denoted with V in Pseudocode 2). This variability is summarized with the interval given by the 5% and the 95% quantiles of the values in V. Finally, the uncertainty introduced in step (4) is assessed by calculating a 90% prediction interval based on bootstrap methods.

Estimates of the prediction uncertainties of the gap filled values could be derived by combining the uncertainties of all four previously described steps. However, doing so in a meaningful way is not straightforward due to possible interactions and elimination effects among them. Nevertheless, we combine the uncertainties from step (2) and (3) in one prediction interval. This prediction interval reflects the local spatial and temporal heterogeneity of the data around the predicted value. An evaluation of the properties of that interval and its practical relevance is given in Section III-C2 and IV-B.

D. Software

The described gap filling approach is implemented in the programming languages R/C++ and is available as open-source R package at http://cran.r-project.org/package=gapfill [60], [61]. The program features a flexible design allowing the user to optimize the gap filling method for specific data sets and to construct new gap filling methods based on the subset-predict framework with little effort. Examples implementing different subset and prediction strategies are given in the reference manual of the R package *gapfill* [60].

In general, satellite products comprise large amounts of data. Therefore, a gap filling method of practical significance has to be efficient in terms of computation resources and scalable in the sense that the gap filling task can be distributed to several computing units. Since the presented method fills each missing value separately by taking only a subset of the data into account, the gap filling task is easily parallelized. We use tools from the R package *foreach* [62] allowing the user to chose between an OpenMP [63] and a MPI [64] back-end to executed code in parallel. More information about the usage and implementation of the R package is given in Section S1 of the supplementary material (http://user.math.uzh.ch/furrer/download/tgrs/supplementary_material.pdf) and in the reference manual of the R package [60].

III. Method and Data for Evaluation

Several test scenarios were constructed and the predicted values, together with their uncertainty components, were investigated. In addition, the accuracy of the filled values was compared against those of two alternative gap filling methods.

A. Data

We considered the MODIS satellite product (MOD13A1), which is part of the MODIS vegetation index product (MOD13) [65]. It is a land surface product based on pre-composited 8-day MODIS Level-2G surface reflectance data, which have been further composited to obtain the final resolution of circa 16 days and 500 m [11], [9]. The NDVI layer can

be used to describe vegetation activity [66]. We used the pixel reliability layer to subset the NDVI data to values flagged as "good data" [10].

To illustrate the gap filling method, NDVI data from the years 2004 to 2009 were considered and restricted to the region of northern Alaska as shown in Fig. 2. Due to the high latitudes ranging from 66° north to more than 71° north, the NDVI values exhibit a strong seasonal component. That is reflected in both the NDVI values and the number of available values classified as "good data". Especially during wintertime, little data are available because of missing sunlight and snow cover. Therefore, we restrict the analysis to the seasonal period starting on the day of the year (DOY) 145 (about May 24) and ending on DOY 257 (about September 13) featuring 8 dates with observations per season. With that, the data of each considered day of the year have at least 30% of the values classified as "good data".

The MOD13A1 data were downloaded in 6 spatial tiles and merged to one single image per considered point in time using the R package *MODIS* [67], which interfaces the MODIS reprojection tool [68]. Furthermore, the data were transformed from the sinusoidal to the geographic map projection (WGS84). The R packages *raster* [69], *sp* [70], [71], *fields* [72], *lattice* [73], and *ggplot2* [74] were used to handle and visualize the data.

B. Test Scenarios

To study the performance of the gap filling software, we constructed four tests scenarios based on the MOD13A1 data. Using real data, as opposed to simulated data, has the advantage that the scenarios come close to the use-case of interest. The scenarios were built by extracting a 100×100-pixel subset of the data described in Section III-A, while keeping the temporal structure of 6 years and 8 seasonal time points per year unchanged. The geographical location of the subset is depicted in Fig. 2 as the rectangle labeled with "Data" and we refer to it as "data subset". It was selected such that it has relatively few missing values (about 12%) and reflects typical features of NDVI data sets in high latitudes. Two of these features are the latitudinal gradient manifesting itself through lower NDVI values in the northern regions and low NDVI values that are caused by surface water. By artificially removing values from the data subset (setting them to NA), test scenarios were retrieved. In this way, we know most of the actually observed values of the test scenarios (denoted with "validation values"), and can compare the values from the gap filled versions of the test scenarios against them. To mimic realistic spatio-temporal patterns of missing values, the removal of NDVI values from the data subset was performed according to patterns of missing values observed at other locations of the Alaska data set (as opposed to remove values randomly). We choose the pattern of missing values observed at the regions of the rectangles that are denoted with "20%", "30%", "40%", and "50%" in Fig. 2. By removing these spatio-temporal patterns of missing values from the data subset, four test scenarios exhibiting 20%, 30%, 40%, and 50% missing values were retrieved. The NDVI values of the test scenarios and summary figures thereof are depicted in Figs. S1–S5 (available online). A further set of test scenarios was created by taking the same data subset and removing values (setting them to NA) randomly. Using this method, test scenarios exhibiting 20%, 30%, 40%, and 50% missing values were retrieved.

In addition, a scenario consisting of the entire spatial extent of northern Alaska, as shown in Fig. 2, was compiled. While the temporal dimensions of that scenario remained unchanged, the size of the imagery is increased to $271'819$ pixels. $3'696'691$ (28%) of the values in that scenario are missing. The scenario is shown in Fig. S13 (available online).

C. Evaluation of the Gap Filling Method

1) Prediction Accuracy: A first evaluation criterion for gap filling methods is the proportion of values that remained missing after applying them. Good gap filling methods are capable of predicting many missing values of a data set. A second criterion is the visual examination of the filled values, which helps to detect artificial patterns introduced by the gap filling method. More objective measurements of the prediction accuracy can be made when considering the fill values of the test scenarios described in Section III-B. Here most of the fill values have a corresponding validation value, and hence, the prediction accuracy can be quantified with the root mean squared error (RMSE) and the mean absolute error (MAE) [75]. They are defined as $\left(\sum_{i=1}^{n}(\hat{z}_i - z_i)^2/n\right)^{1/2}$, and $\sum_{i=1}^{n}|\hat{z}_i - z_i|/n$, respectively, where n is the number of predicted values $\hat{z}_i$ and where z_i denote the corresponding validation values. Furthermore, we investigated the distribution as well as spatial and temporal patterns of the RMSE.

2) Uncertainty Assessment: The gap filling method exhibits four main steps that are linked to the uncertainties of the predicted values (Section II-C). To compare the magnitude of these individual uncertainty contributions, the lengths of the corresponding prediction intervals are summarized. This investigation was performed using the values of DOY 145 and 161 from the test scenario with 40% missing values. The properties of the 90% prediction interval combining the uncertainties from step (2) and (3) were assessed by investigating the spatial and temporal distribution of the prediction interval lengths and by calculating the coverage rate of the intervals, i.e., the proportion of intervals that cover the observed value (assumed to be true). This part of the uncertainty assessment was performed using the entire test scenario with 40% missing values.

3) Comparison with TIMESAT and Gapfill-Python: We compared the results from the described gap filling method—denoted as the corresponding R package with "*gapfill*"—against two alternative methods, namely the gap filling method presented in [46] and the temporal interpolation methods implemented in the software TIMESAT [13]. The former belongs to the class of methods that reconstruct a complete data product from the observed values (Section I-B2). It applies one of two different prediction algorithms depending on the amount of missing values and exploits both the temporal and the spatial structure of the data. A python notebook is

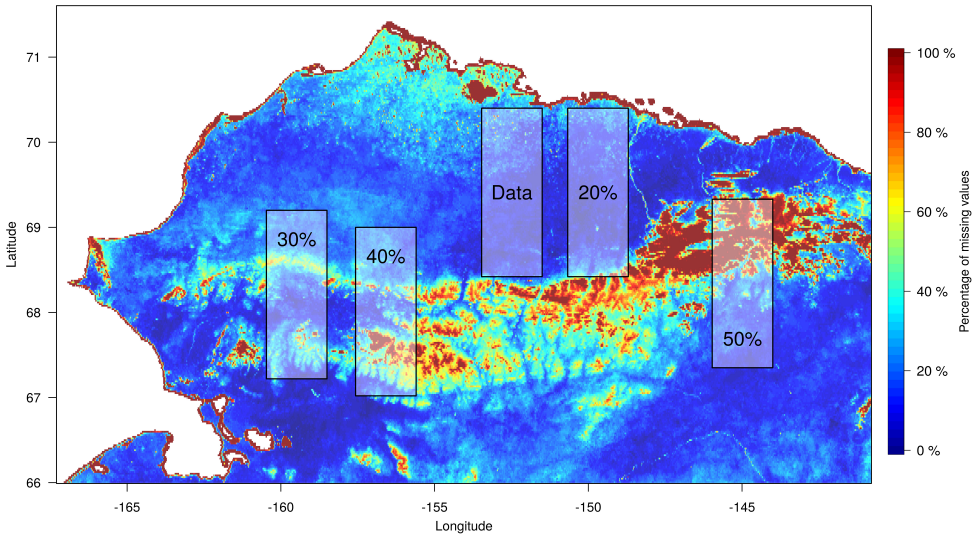


Fig. 2. A map of the study region of northern Alaska. The colors indicate the percentage of missing values of all 48 considered time points. To construct test scenarios, we consider the NDVI values of the 100×100-pixels region labeled with "Data". That subset exhibits relatively few (about 12%) missing values. Test scenarios with 20%, 30%, 40%, and 50% missing values were obtained by artificially removing values from the "Data" region according to the patterns of missing values observed at the 100×100-pixels regions labeled with "20%", "30%", "40%", and "50%", respectively. Note that the 100×100-pixels regions are depicted as rectangles, as opposed to squares, because of the chosen geographic map projection.

available at github.com/malaria-atlas-project/modis-gapfilling and provides an implementation of the method in Python [76] and C [77]. The code was downloaded on August 15, 2015 (git commit c83776c). In the following, we will refer to that software as "Gapfill-Python". While that method was published in 2014, the TIMESAT software is more than ten years older and well established. It is implemented in Fortran and comes with a MATLAB [78] interface featuring a graphical user interface. The software (version 3.2) and the documentation thereof is available at web.nateko.lu.se/timesat/. The main purpose of the software is to analyze time series of satellite data by extracting seasonal parameters from a smoothed version of the time series. All calculations treat the pixel-wise time series separately, and hence, do not exploit the spatial dependency in the data. The smoothing part of the method makes the analysis, to some extent, robust to outliers and missing values. The method therefore belongs to the class of methods that handle missing values through robust analysis techniques (Section I-B1). Although the software was designed to use the smoothed time series in conjunction with the extraction of phenological parameters, the smoothing part of the algorithm can be used for gap filling and the gap filled time series were used to assess the performance of different types of smoothers [19], [22], [23].

All considered gap filling methods have several tuning parameters, which influence the prediction process and the accuracy of the predictions. Although, we tried to find good parameter configurations for the presented software, it may be that the results improve with other settings. Nevertheless, the presented comparisons give a solid overview of the performance. The tuning parameters for the *gapfill* method are given in Section II-A. The R-code to execute the *gapfill* program is available in Listing S1 (available online). Gapfill-Python has about 16 parameters to be set. The most important ones are those controlling the search of informative values and the "de-speckle" algorithm (see the Python-notebook for more information). The used parameter settings are given in Listing S2 (available online). The parameters of TIMESAT are described in its software manual [79]. We chose to fit a "double logistic" smoothing function, which is recommended for NDVI values in high latitudes with many missing values [80] and provided the most satisfying results. The complete configuration file shown in Listing S3 (available online).

To compare the predicted values from *gapfill*, Gapfill-Python, and TIMESAT, we applied the methods to the four test scenarios and report the number and proportion of successfully filled values as well as the RMSE and the MAE of the predicted values. We used two-sided paired two-sample Wilcoxon tests with a significance level $\alpha = 0.05$ to assess whether the MAE and RMSE from Gapfill-Python and TIMESAT are different from those of *gapfill* [81]. In addition, we compared the distribution as well as the spatial and temporal patterns of the difference between the absolute errors (AEs) of *gapfill* and Gapfill-Python.

IV. Results

A. Prediction Accuracy

We applied the proposed gap filling method to the test scenarios described in Section III-B. It returned predictions for all missing values in all test scenarios. Images of the resulting gap filled data sets are shown in Figs. S6–S9 (available online) and Fig. 3 (left), which depict the predicted values of DOY 177 in 2006 for the test scenario with 40% missing values. A visual examination of the gap filled images did not reveal any

artificially introduced spatial pattern. Moreover, the images reconstruct the spatial distribution of the NDVI values well; this includes small-scale features such as, e.g., the band of low NDVI values crossing the image from the west to the east, which is present in all images. Time-series of three pixels from the test scenario with 40% missing values are shown in Fig. 4. The pixels were selected such that they represent locations with large, average, and low NDVI values. The predicted values follow the expected seasonal curve and match the validation values well. Low NDVI values are more difficult to predict (bottom panel of Fig. 4) as they have larger uncertainties (Fig. S1, available online).

To assess the temporal variation of the prediction accuracy, the mean RMSEs per time point of the scenario with 40% missing values are shown in the middle panel of Fig. 3. It can be seen that the RMSE is larger for early days of the year. This is in accordance with the observation that values at the beginning of the season are more likely to be missing and exhibit larger variability compared to later observations (Fig. S1, available online). The right panel of Fig. 3 depicts the spatial distribution of the mean pixel-wise RMSEs, which resembles the spatial distribution of the temporal variation in the data (top right panel of Fig. S1, available online). This is expected, because values of pixels with a large variability in time are more difficult to predict.

Another way to study biases and the variability of the predicted values is to plot the validation values against the predicted values as shown in the upper panels of Fig. 5. Most of the validation values are between 0.3 and 0.8. In that interval the predicted values are scattered around the diagonal (red line) indicating that they are near the validation values on average. Pixels with values below 0.5 have lower prediction accuracy. This is in accordance with the observation that those pixels tend to have larger variance and are therefore naturally more difficult to predict. As expected, the deviation of the predicted values from the validation values increases with larger percentages of missing values. This can also be seen in the bottom panels of Fig. 5, where the histograms of the prediction errors (predicted minus validation values) show a wider distribution with increasing percentages of the missing values. While the medians of the error differences are located at zero, the distributions of the differences are positively skewed (skewness between 2.17 and 2.3). This is a shrinkage effect reflecting an increased prediction uncertainty for low NDVI values. The standard deviation of the distributions of differences is between 0.0413 and 0.0421 for the test scenario with 20%, 30%, and 40% missing values, and increases to 0.0588 for the test scenario with 50% missing values. This increase could be due to an increased amount of low NDVI values in the test scenario with 50% missing values. These low NDVI values tend to have a large variability over time (Fig. S1, available online) and are therefore more difficult to predict.

In addition, *gapfill* was applied to the entire spatial region of northern Alaska depicted in Fig. 2. The images of the gap filled data are shown in Fig. S14 (available online). Again, all missing values were filled and a visual inspection of the data did not reveal any artificially introduced patterns.

B. Uncertainty Assessment

The widths of the prediction intervals, corresponding to the four main steps of the gap filling method, summarize their uncertainty contribution. The left panel of Fig. 6 depicts summary statistics of these widths as boxplots revealing that the sorting step (2) (Pseudocode 1) introduced the largest uncertainties, followed by the estimation of the quantile of step (3) (Pseudocode 2). To investigate the properties of the prediction interval combining the uncertainties from step (2) and (3), the spatial distribution of the mean pixel-wise prediction interval widths is shown in the middle panel of Fig. 6. It exhibits similar spatial patterns as the standard deviation of the data (top right panel of Fig. S1, available online) and the spatial distribution of the average RMSEs (right panel of Fig. 3). Since the seasonal variability of the prediction interval widths is larger, compared to the inter-annual variability, we only show the former in the right bottom panel of Fig. 6. It has a U-shape, which is also observed in the distributions of the missing values (Fig. S1, available online) with some deviations early and late in the season, i.e., the values of DOY 145 and 257. These deviations might be caused by the fact that we only consider a part of the seasonal cycle, and, hence, have less information at the boundaries thereof. The overall coverage rate of the prediction interval for that scenario is 93%. This is, the prediction uncertainty is slightly overestimated on average. The average coverage rate per day of the year is depicted in the right panels of Fig. 6.

C. Comparison with TIMESAT and Gapfill-Python

When comparing the predictions of *gapfill* with those from Gapfill-Python and TIMESAT, one performance measurement of interest is the ability to fill gaps in scenarios exhibiting many missing values. To investigate that, we calculated the number of (non-NA) fill values and the percentage of (non-NA) fill values relative to the total number of missing values; see, columns "# filled" of Tab. II. While *gapfill* predicted all missing values of the four test scenarios, Gapfill-Python and TIMESAT returned NA predictions for some values. The number of NAs in the predictions seems to increase with the number of NAs in the input data and are a considerable proportion (up to 94%) for the TIMESAT predictions. The large amount of missing values in the TIMESAT predictions might be explained by the uneven distribution of missing values in the data. This is, a large share of the missing values occurs in time series with many missing values. In addition, we considered only data from 8 of the 24 time points of the seasonal cycle, because the quality of the values from the remaining 16 winter time points was too low. This makes it challenging for a time series based approach like TIMESAT to predict missing values.

The RMSE for *gapfill* is between 41.34×10^{-3} and 42.54×10^{-3} for the test scenarios with 20%, 30%, and 40% missing values, and increases to 59.58×10^{-3} for the test scenario with 50% missing values. This increase is not surprising, since the difficulty of predicting missing values depends not only on the percentage of missing values, but also on the distribution thereof, which is different in all test scenarios.

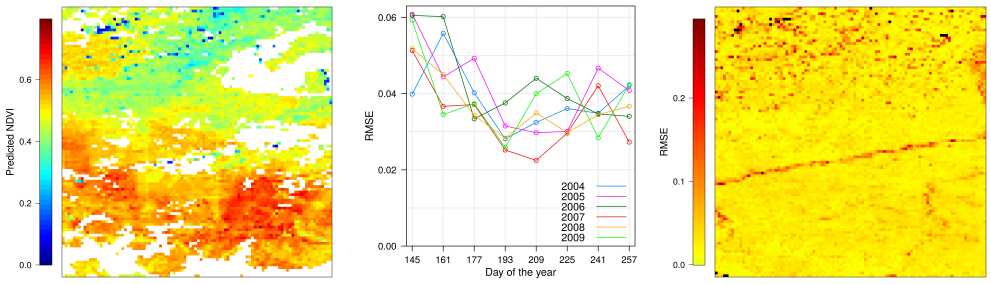


Fig. 3. Predictions and accuracy measurements for the test scenario with 40% missing values. Left: predicted NDVI values for the day 177 of the year 2006. For that image 2′335 of 10′000 values were observed and are depicted as white pixels. Middle: RMSE for the indicated dates. Right: spatial distribution of the RMSE. 19 RMSE values are missing because these pixels were observed in all time points. They are depicted as black pixels. Note that the 100×100-pixels images (left and right panel) are shown as squares having the latitude on the y-axis and longitude on the x-axis, respectively.

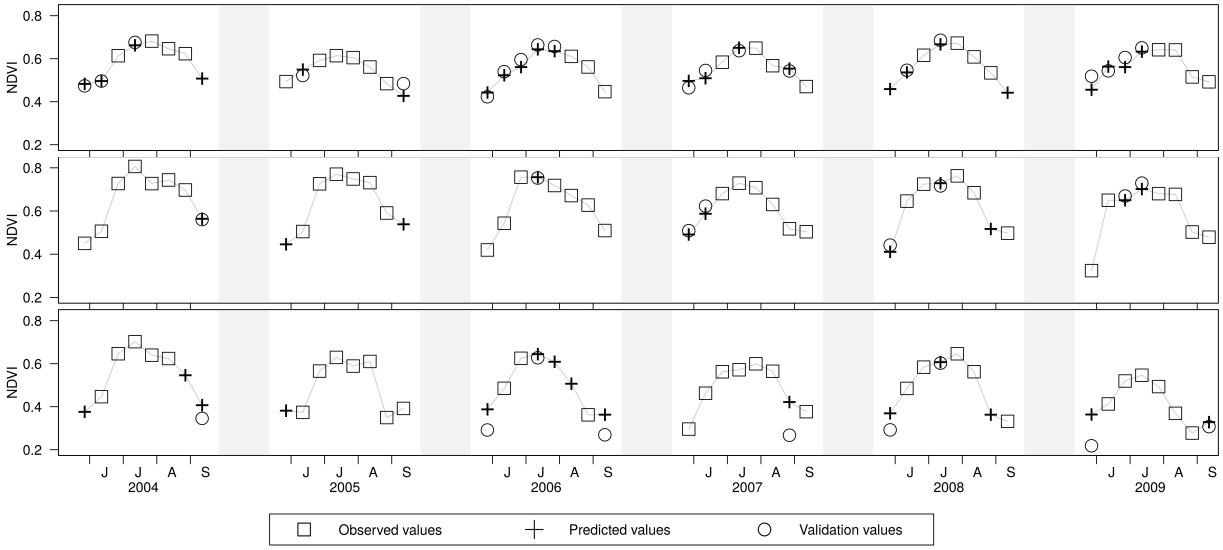


Fig. 4. Temporal profiles for three pixels of the scenario with 40% missing values. The pixels were selected such that they represent locations with large (top), average (middle), and low (bottom) NDVI values. If validation values were available, they are displayed with circles. The gray areas represent the time span from mid September to mid May with low vegetation activity.

TABLE II
THE GAP FILLED VALUES OF THE FOUR TEST SCENARIOS OBTAINED WITH *gapfill*, GAPFILL-PYTHON AND TIMESAT ARE SUMMARIZED IN TERMS OF THE NUMBER AND PERCENTAGE OF FILLED VALUES AND THE RMSE $\times 10^3$. TO GET COMPARABLE RESULTS, THE RMSES OF *gapfill* ARE ALSO GIVEN FOR THE SUBSETS WITH AVAILABLE FILLED VALUES FROM GAPFILL-PYTHON (RMSE_P) AND TIMESAT (RMSE_P)

	gapfill				Gapfill-Python		TIMESAT	
	#filled	RMSE	RMSE_P	RMSE_T	#filled	RMSE	#filled	RMSE
20%	92’822 (100%)	41.80	42.06	41.10	90’307 (97%)	45.00	59’948 (65%)	83.43
30%	147’827 (100%)	42.54	42.39	37.09	146’686 (99%)	45.54	42’892 (29%)	71.43
40%	192’456 (100%)	41.34	40.98	36.41	169’998 (88%)	42.49	31’279 (16%)	71.93
50%	240’326 (100%)	59.58	44.94	37.24	134’540 (56%)	45.61	14’127 (6%)	86.09

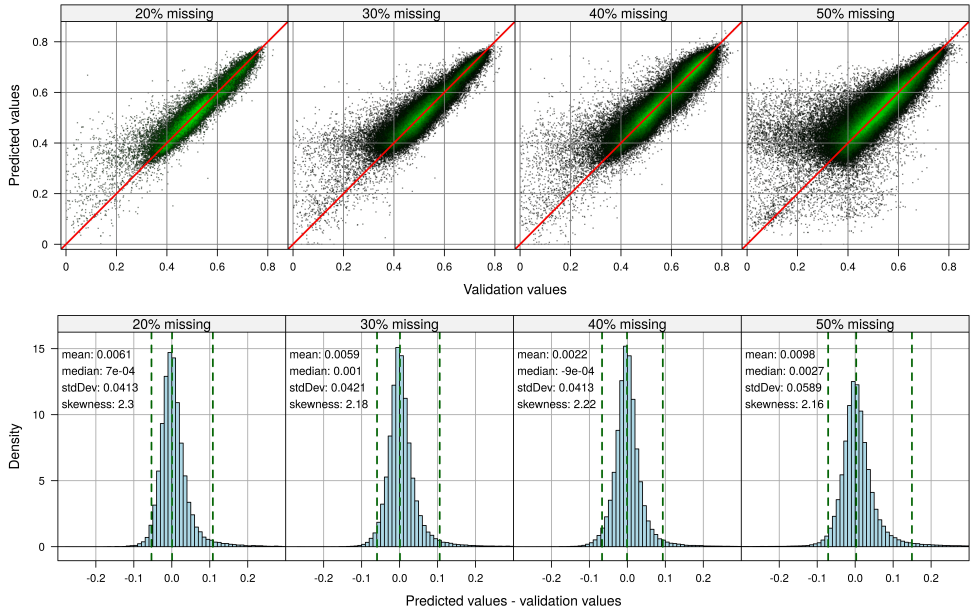


Fig. 5. Accuracy of the *gapfill* predictions for the four test scenarios of the validation study. Upper panels: scatter plots of the predicted values (y-axis) versus the validation values (x-axis). The green color shading indicates regions with a large density of points; light green corresponds to 100 overlaying points. (A similar figure for the TIMESAT and the Gapfill-Python method discussed in Section IV-C is given in Fig. S16 of the supplementary material available online.) Bottom panels: histograms of the differences between the predicted and the validation values. The dashed green lines indicate the 2.5%, 50%, and 97.5% quantiles.

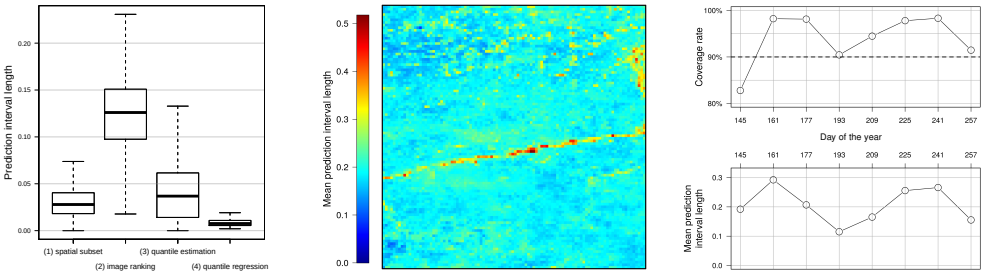


Fig. 6. Left: uncertainty contribution from the indicated four steps of the gap filling method. Middle: spatial distribution of the mean pixel-wise width of the 90% prediction intervals for the test scenario with 40% missing values. The intervals are based on the uncertainties from step (2) (Pseudocode 1) and step (3) (Pseudocode 2). Right: the corresponding coverage rates and mean interval widths per day of the year.

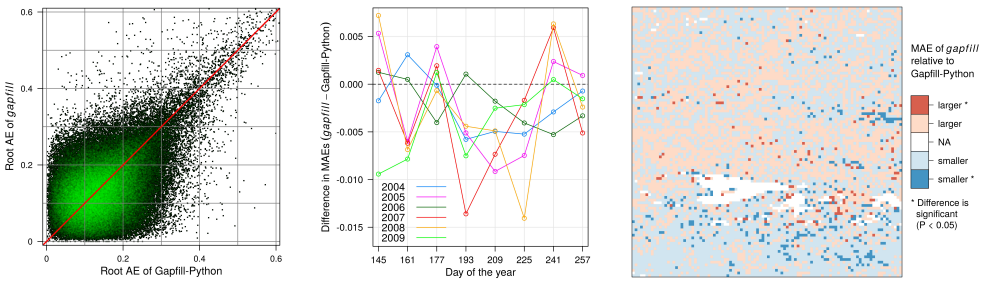


Fig. 7. Comparison of the AEs from *gapfill* and Gapfill-Python for the test scenario with 40% missing values. Left: scatter plot of the root AEs of *gapfill* (y-axis) and Gapfill-Python (x-axis). The green color shading indicates regions with a large density of points (light green corresponds to 25 overlaying points). Middle: difference of the MAEs of *gapfill* and Gapfill-Python for the indicated dates. Right: Spatial comparison of the MAEs. Colors indicate whether the MAE of *gapfill* or Gapfill-Python is larger. Dark colors indicate that the differences are significant. For the 385 white pixels no comparison was made because all values for the corresponding pixels were observed or Gapfill-Python did not fill the pixels.

Also Gapfill-Python and TIMESAT have more problems to gap fill the test scenario with 50% missing values; compare the increase in the proportions of not filled values in Tab. II. The prediction accuracy in terms of the RMSE can only be calculated for non-missing predictions. Thus, the RMSE of the Gapfill-Python and the TIMESAT predictions are based on a subset of the values to predict only. To make the RMSEs of those methods comparable to the RMSEs of *gapfill*, we also calculated the RMSEs of the *gapfill* method relative to the subsets of predicted values from Gapfill-Python (RMSE_P) and TIMESAT (RMSE_P), respectively. The RMSEs given in Tab. II indicate that the *gapfill* predictions are the most accurate ones in all scenarios. The MAEs show a similar pattern as the RMSEs (Tab. S1, available online). Both the RMSEs and MAEs of *gapfill* are significantly smaller than those from Gapfill-Python and TIMESAT for all test scenarios (Wilcoxon tests, all p-values $< 10^{-15}$). In addition, we created tables similar to Tab. II and Tab. S1 for the test scenarios with randomly removed values. The obtained RMSEs and MAEs are shown in Tab. S2 and Tab. S3 (available online). Both error measures indicate that the fill values of *gapfill* are the most accurate ones (Wilcoxon tests, all p-values $< 10^{-15}$).

While the RMSEs of Gapfill-Python are close to those of *gapfill*, the RMSEs of TIMESAT are about two times higher compared to those of *gapfill*. We therefore restrict the following comparison to the *gapfill* and the Gapfill-Python methods and investigate the test scenario with 40% missing values in more detail. The left panel of Fig. 7 depicts a scatter plot of the root AEs of *gapfill* (y-axis) and Gapfill-Python (x-axis). The root transformation was chosen to facilitate the visual inspection of small differences. In this figure the scattering seems to be symmetric around the red line and no clear pattern discriminates the methods. In the middle panel of Fig. 7, the differences of the MAEs of *gapfill* and Gapfill-Python are shown for all time points. For 33 of the 48 time points the *gapfill* method performed better. Despite of that no clear temporal pattern can be seen. Finally, a spatial comparison of the methods is given in the right panel of Fig. 7. In that panel, the 100×100-pixel area of the test scenario is depicted. The colors indicate whether the MAEs of *gapfill* are larger or smaller compared to that of Gapfill-Python. Opaque colors indicate that the differences are significant (Wilcoxon tests, $\alpha = 0.05$). For 372 (72.2%) of the total 508 pixels exhibiting significant differences, the *gapfill* method performed better. Especially in the southern region the *gapfill* method seems to perform better, whereas the other regions do not show a clear pattern. This could be due to the accumulation of missing values in that area (Fig. S4, available online). Figures of the complete spatio-temporal pattern of the AEs of *gapfill* and Gapfill-Python as well as the difference thereof are given in Figs. S10–S12 (available online).

V. Discussion

The analysis of remotely sensed data with many missing observations is challenging. One way to handle missing values is to construct a complete data set by predicting the missing values from the observed ones. Such an approach is presented in this study and implemented in the corresponding R package *gapfill*. The following four considerations influenced the development of the gap filling method:

Firstly, to be of practical relevance, the method has to be capable of handling large data sets, such as, e.g., MODIS NDVI products. This implies that classical geostatistical space-time models in the spirit of [82] would need severe modifications, since their computation workload typically exceeds the available resources by several orders of magnitude. One way to reduce the computational workload is to implement a purely algorithmic gap filling method as, e.g., presented by [46]. While such approaches can achieve good performance in terms of prediction accuracy, uncertainty quantification is difficult. We therefore opted for a hybrid approach, which combines purely algorithmic elements (selection of a suitable subset; scoring of images) together with statistical methods (quantile regression; permutation tests). With that, the gap filling method benefits from both aspects: The algorithmic components make it fast and scalable, whereas the statistics part provides tools to quantify uncertainties.

Secondly, an efficient software implementation of the gap filling method is crucial to handle large data sets. Since nowadays many research institutes have access to powerful computers, this includes scalability of the method so that the computational workload can be distributed among several computing units. For example, we gap filled the $\approx 3.7 \times 10^6$ missing values of the Alaska NDVI data set introduced in Section III using 80 cores of an Intel® Xeon® CPU E7-2850 @ 2.00 GHz in circa 10 hours. The computation time reduces (almost) linearly with the number of used cores. This scalability is achieved with the subset-predict framework, which handles each missing value separately and thereby enables parallelization. On the programming side, we employ the generic parallelization framework of the R package foreach [62], which can be used to interface either openMP or MPI depending on the architecture of the available computer.

Thirdly, the proposed gap filling software should be flexible and user-friendly so that it can be tailored to specific features of different remote sensing data products. To that end, we provide an implementation of the method as open-source R package *gapfill* [60], which contains the R/C++ source code together with documentation and data examples. The structure of the package was kept as simply as possible to ease its usage. On the other hand, users can customize the essential parts of the method by changing default parameters or by providing their own subset and/or predict functions; see the supplementary material available online.

Fourthly, testing the *gapfill* software with realistic scenarios was essential to develop an accurate and fast method. The chosen test scenarios feature actually observed MODIS NDVI data together with observed patterns of missing values and are therefore close to an actual use-case. In the development phase of the gap filling method, testing helped to detect alleged improvements, which turned out to be deteriorations with respect to computational speed or prediction accuracy. Also, the comparison against established software provides valuable information for potential users. However, one needs to keep in mind that such comparisons are always relative to the choice

of the test scenarios.

VI. Conclusion

Since many analysis methods for remotely sensed data are designed to make use of spatially and temporally continuous data, gap filling missing values improves or enables data analysis in the presence of missing values. We presented such a gap filling method and provide an open-source implementation thereof in the R package *gapfill*. While the software readily works for the presented data, its flexible design allows users to tailor it to specific needs with little effort. With that it is a suitable tool to gap fill many data products observed at regularly spaced points in time. Furthermore, the method has statistical components, which enable rigorous uncertainty quantification. The performance of the predicted values was assessed using test scenarios based on MODIS NDVI data featuring between 20% and 50% missing values. In contrast to two alternative gap filling methods, the presented method was able to fill all missing values in the test scenarios. Moreover, the RMSEs for the test scenarios were between 0.041 and 0.060, which was lower compared to those from the two alternative methods (Wilcoxon tests, all $p < 10^{-15}$).

Acknowledgment

We thank Dr. H. Gibson for sharing the Gapfill-Python software and for answering related questions. Moreover, we thank two anonymous reviewers for their helpful comments.

References

[1] International Satellite Cloud Climatology Project (ISCCP), 2016. [Online]. Available: http://isccp.giss.nasa.gov/products/browsed2.html

[2] D. A. Stow, A. Hope, D. McGuire, D. Verbyla, J. Gamon, F. Huemmrich, S. Houston, C. Racine, M. Sturm, K. Tape, L. Hinzman, K. Yoshikawa, C. Tweedie, B. Noyle, C. Silapaswan, D. Douglas, B. Griffith, G. Jia, H. Epstein, D. Walker, S. Daeschner, A. Petersen, L. Zhou, and R. Myneni, "Remote sensing of vegetation and land-cover change in Arctic Tundra ecosystems," *Remote Sens. Environ.*, vol. 89, no. 3, pp. 281–308, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425703002803

[3] G. P. Asner, "Cloud cover in Landsat observations of the Brazilian Amazon," *Int. J. Remote Sens.*, vol. 22, no. 18, pp. 3855–3862, 2001.

[4] G. Hmimina, E. Dufrêne, J.-Y. Pontailler, N. Delpierre, M. Aubinet, B. Caquet, A. de Grandcourt, B. Burban, C. Flechard, A. Granier, P. Gross, B. Heinesch, B. Longdoz, C. Moureaux, J.-M. Ourcival, S. Rambal, L. S. André, and K. Soudani, "Evaluation of the potential of MODIS satellite data to predict vegetation phenology in different biomes: An investigation using ground-based NDVI measurements," *Remote Sens. Environ.*, vol. 132, pp. 145–158, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425713000229

[5] I. Garonna, R. de Jong, and M. E. Schaepman, "Variability and evolution of global land surface phenology over the past three decades (1982–2012)," *Glob. Chang. Biol.*, vol. 22, no. 4, pp. 1456–1468, 2016.

[6] M. A. White, K. M. De Beurs, K. Didan, D. W. Inouye, A. D. Richardson, O. P. Jensen, J. O'Keefe, G. Zhang, R. R. Nemani, W. J. D. Van Leeuwen, J. F. Brown, A. De Wit, M. Schaepman, X. Lin, M. Dettinger, A. S. Bailey, J. Kimball, M. D. Schwartz, D. D. Baldocchi, J. T. Lee, and W. K. Lauenroth, "Intercomparison, interpretation, and assessment of spring phenology in North America estimated from remote sensing for 1982–2006," *Glob. Chang. Biol.*, vol. 15, no. 10, pp. 2335–2359, 2009.

[7] J. v. Buttlar, J. Zscheischler, and M. D. Mahecha, "An extended approach for spatiotemporal gapfilling: Dealing with large and systematic gaps in geoscientific datasets," *nonlinear Proc. Geoph.*, vol. 21, no. 1, pp. 203–215, 2014.

[8] C. Gómez, J. C. White, and M. A. Wulder, "Optical remotely sensed time series data for land cover classification: A review," *ISPRS J. Photogramm. Remote Sens.*, vol. 116, pp. 55–72, 2016.

[9] W. J. van Leeuwen, A. R. Huete, and T. W. Laing, "MODIS vegetation index compositing approach: A prototype with AVHRR data," *Remote Sens. Environ.*, vol. 69, no. 3, pp. 264–280, 1999.

[10] D. P. Roy, J. S. Borak, S. Devadiga, R. E. Wolfe, M. Zheng, and J. Descloitres, "The MODIS land product quality assessment approach," *Remote Sens. Environ.*, vol. 83, no. 1–2, pp. 62–76, 2002, the Moderate Resolution Imaging Spectroradiometer (MODIS): a new generation of Land Surface Monitoring.

[11] K. Didan, A. B. Munoz, R. Solano, and A. Huete, *MODIS vegetation index user's guide (MOD13 Series)*, 2015, version 3.00 (Collection 6). [Online]. Available: http://vip.arizona.edu/documents/MODIS/MODIS_VI_UsersGuide_June_2015_C6.pdf

[12] J. Chen, P. Jönsson, M. Tamura, Z. Gu, B. Matsushita, and L. Eklundh, "A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky-Golay filter," *Remote Sens. Environ.*, vol. 91, no. 3-4, pp. 332–344, 2004.

[13] P. Jönsson and L. Eklundh, "TIMESAT—a program for analyzing time-series of satellite sensor data," *Comput. Geosci.*, vol. 30, no. 8, pp. 833–845, 2004.

[14] F. Gao, J. Morisette, R. Wolfe, G. Ederer, J. Pedelty, E. Masuoka, R. Myneni, B. Tan, and J. Nightingale, "An algorithm to produce temporally and spatially continuous MODIS-LAI time series," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 1, pp. 60–64, 2008.

[15] G. J. Roerink, M. Menenti, and W. Verhoef, "Reconstructing cloudfree NDVI composites using Fourier analysis of time series," *Int. J. Remote Sens.*, vol. 21, no. 9, pp. 1911–1917, 2000.

[16] J. P. W. Scharlemann, D. Benz, S. I. Hay, B. V. Purse, A. J. Tatem, G. R. W. Wint, and D. J. Rogers, "Global data for ecology and epidemiology: A novel algorithm for temporal Fourier processing MODIS data," *PLoS ONE*, vol. 3, no. 1, p. e1408, 2008.

[17] A. Moreno, F. J. Garcia-Haro, B. Martinez, and M. A. Gilabert, "Noise reduction and gap filling of fAPAR time series using an adapted local regression filter," *Rem. Sens.*, vol. 6, no. 9, p. 8238, 2014.

[18] M. Neteler, "Estimating daily land surface temperatures in mountainous environments by reconstructed MODIS LST data," *Rem. Sens.*, vol. 2, no. 1, pp. 333–351, 2010.

[19] A. Verger, F. Baret, M. Weiss, S. Kandasamy, and E. F. Vermote, "The CACAO method for smoothing, gap filling, and characterizing seasonal anomalies in satellite time series." *IEEE T. Geoscience and Remote Sensing*, vol. 51, no. 4-1, pp. 1963–1972, 2013.

[20] R. Colditz, C. Conrad, T. Wehrmann, M. Schmidt, and S. Dech, "TiSeG: A flexible software tool for time-series generation of MODIS data utilizing the quality assessment science data set," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 10, pp. 3296–3308, 2008.

[21] J. P. Musial, M. M. Verstraete, and N. Gobron, "Technical note: Comparing the effectiveness of recent algorithms to fill and smooth incomplete and noisy time series," *Atmos. Chem. Phys.*, vol. 11, no. 15, pp. 7905–7923, 2011.

[22] P. M. Atkinson, C. Jeganathan, J. Dash, and C. Atzberger, "Inter-comparison of four models for smoothing satellite sensor time-series data to estimate vegetation phenology," *Remote Sens. Environ.*, vol. 123, pp. 400–417, 2012.

[23] J. N. Hird and G. J. McDermid, "Noise reduction of NDVI time series: An empirical comparison of selected techniques," *Remote Sens. Environ.*, vol. 113, no. 1, pp. 248–258, 2009.

[24] S. Kandasamy, F. Baret, A. Verger, P. Neveux, and M. Weiss, "A comparison of methods for smoothing and gap filling time series of remote sensing observations; application to MODIS LAI products," *Biogeosciences*, vol. 10, no. 6, pp. 4055–4071, 2013.

[25] C. Atzberger and P. H. C. Eilers, "Evaluating the effectiveness of smoothing algorithms in the absence of ground reference measurements," *Int. J. Remote Sens.*, vol. 32, no. 13, pp. 3689–3709, 2011.

[26] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sens. Environ.*, vol. 114, no. 1, pp. 106–115, 2010.

[27] J. Verbesselt, R. Hyndman, A. Zeileis, and D. Culvenor, "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series," *Remote Sens. Environ.*, vol. 114, no. 12, pp. 2970–2980, 2010.

[28] S. Jamali, P. Jönsson, L. Eklundh, J. Ardö, and J. Seaquist, "Detecting changes in vegetation trends using time series segmentation," *Remote Sens. Environ.*, vol. 156, pp. 182–195, 2015.

[29] A. Verger, F. Baret, and M. Weiss, "Near real-time vegetation monitoring at global scale," *IEEE J. Sel. Topics Appl. Earth Observ. in Remote Sens.*, vol. 7, no. 8, pp. 3473–3481, 2014.

[30] D. Bolin, J. Lindström, L. Eklundh, and F. Lindgren, "Fast estimation of spatially dependent temporal vegetation trends using Gaussian Markov random fields," *Comput. Stat. Data Anal.*, vol. 53, no. 8, pp. 2885–2896, 2009.

[31] Y. Zhu, E. L. Kang, Y. Bo, Q. Tang, J. Cheng, and Y. He, "A robust fixed rank kriging method for improving the spatial completeness and accuracy of satellite SST products," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 9, pp. 5021–5035, 2015.

[32] E. Addink, "A comparison of conventional and geostatistical methods to replace clouded pixels in NOAA-AVHRR images," *Int. J. Remote Sens.*, vol. 20, no. 5, pp. 961–977, 1999.

[33] R. E. Rossi, J. L. Dungan, and L. R. Beck, "Kriging in the shadows: Geostatistical interpolation for remote sensing," *Remote Sens. Environ.*, vol. 49, no. 1, pp. 32–40, 1994.

[34] M. Pringle, M. Schmidt, and J. Muir, "Geostatistical interpolation of SLC-off landsat ETM+ images," *ISPRS J. Photogramm. Remote Sens.*, vol. 64, no. 6, pp. 654–664, 2009.

[35] C. Zhang, W. Li, and D. J. Travis, "Gaps-fill of SLC-off Landsat ETM+ satellite image using a geostatistical approach," *Int. J. Remote Sens.*, vol. 28, no. 22, pp. 5103–5122, 2007.

[36] ——, "Restoration of clouded pixels in multispectral remotely sensed imagery with cokriging," *Int. J. Remote Sens.*, vol. 30, no. 9, pp. 2173–2195, 2009.

[37] L. Guo, L. Lei, Z. C. Zeng, P. Zou, D. Liu, and B. Zhang, "Evaluation of spatio-temporal variogram models for mapping Xco_2 using satellite observations: A case study in China," *IEEE J. Sel. Topics Appl. Earth Observ. in Remote Sens.*, vol. 8, no. 1, pp. 376–385, 2015.

[38] Z. Zeng, L. Lei, S. Hou, F. Ru, X. Guan, and B. Zhang, "A regional gap-filling method based on spatiotemporal variogram model of CO_2 columns," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 6, pp. 3594–3603, 2014.

[39] L. Poggio, A. Gimona, and I. Brown, "Spatio-temporal MODIS EVI gap filling under cloud cover: An example in Scotland," *ISPRS J. Photogramm. Remote Sens.*, vol. 72, pp. 56–72, 2012.

[40] R. Lguensat, P. Tandeo, R. Fablet, and R. Garello, "Spatio-temporal interpolation of sea surface temperature using high resolution remote sensing data," in *2014 Oceans - St. John's*, 2014, pp. 1–4.

[41] N. Golyandina and E. Osipov, "The "Caterpillar"-SSA method for analysis of time series with missing values," *J. Stat. Plan. Inference*, vol. 137, no. 8, pp. 2642–2653, 2007, 5th St. Petersburg Workshop on Simulation.

[42] J. M. Beckers and M. Rixen, "EOF calculations and data filling from incomplete oceanographic datasets," *J. Atmos. Oceanic Technol.*, vol. 20, no. 12, pp. 1839–1856, 2003.

[43] J. Chen, X. Zhu, J. E. Vogelmann, F. Gao, and S. Jin, "A simple and effective method for filling gaps in landsat ETM+ SLC-off images," *Remote Sens. Environ.*, vol. 115, no. 4, pp. 1053–1064, 2011.

[44] M. Mohammdy, H. Moradi, H. Zeinivand, A. Temme, H. Pourghasemi, and H. Alizadeh, "Validating gap-filling of Landsat ETM+ satellite images in the Golestan Province, Iran," *Arabian J. Geosci.*, vol. 7, no. 9, pp. 3633–3638, 2014.

[45] X. Zhu, D. Liu, and J. Chen, "A new geostatistical approach for filling gaps in Landsat ETM+ SLC-off images," *Remote Sens. Environ.*, vol. 124, pp. 49–60, 2012.

[46] D. J. Weiss, P. M. Atkinson, S. Bhatt, B. Mappin, S. I. Hay, and P. W. Gething, "An effective approach for gap-filling continental scale remotely sensed time-series," *ISPRS J. Photogramm. Remote Sens.*, vol. 98, pp. 106–118, 2014.

[47] J. C. de Oliveira, J. C. N. Epiphanio, and C. D. Renno, "Window regression: A spatial-temporal analysis to estimate pixels classified as low-quality in MODIS NDVI time series," *Rem. Sens.*, vol. 6, no. 4, p. 3123, 2014.

[48] J. C. de Oliveira and J. C. N. Epiphanio, "Noise reduction in MODIS NDVI time series data based on spatial-temporal analysis," in *Geoscience and remote sensing symposium (IGARSS), 2012 IEEE International*, 2012, pp. 2372–2375.

[49] S. Kang, S. W. Running, M. Zhao, J. S. Kimball, and J. Glassy, "Improving continuity of MODIS terrestrial photosynthesis products using an interpolation scheme for cloudy pixels," *Int. J. Remote Sens.*, vol. 26, no. 8, pp. 1659–1676, 2005.

[50] S. K. Maxwell, G. L. Schmidt, and J. C. Storey, "A multiscale segmentation approach to filling gaps in Landsat ETM+ SLCoff images," *Int. J. Remote Sens.*, vol. 28, no. 23, pp. 5339–5356, 2007.

[51] J. Gu, X. Li, C. Huang, and G. S. Okin, "A simplified data assimilation method for reconstructing time-series MODIS NDVI data," *Adv. Space Res.*, vol. 44, no. 4, pp. 501–509, 2009.

[52] M. E. Schaepman, M. Jehle, A. Hueni, P. D'Odorico, A. Damm, J. Weyermann, F. D. Schneider, V. Laurent, C. Popp, F. C. Seidel, K. Lenhard, P. Gege, C. Küchler, J. Brazile, P. Kohler, L. D. Vos, K. Meuleman, R. Meynart, D. Schläpfer, M. Kneubühler, and K. I. Itten, "Advanced radiometry measurements and Earth science applications with the Airborne Prism Experiment (APEX)," *Remote Sens. Environ.*, vol. 158, pp. 207–219, 2015.

[53] J. V. Martins, D. Tanré, L. Remer, Y. Kaufman, S. Mattoo, and R. Levy, "MODIS cloud screening for remote sensing of aerosols over oceans using spatial variability," *Geophys. Res. Lett.*, vol. 29, no. 12, pp. MOD4–1–MOD4–4, 2002.

[54] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statist. Surv.*, vol. 4, pp. 40–79, 2010.

[55] F. Mosteller and J. W. Tukey, "Data analysis, including statistics," in *The Collected Works of John W. Tukey: Philosophy and Principles of Data Analysis 1965–1986*, L. V. Jones, Ed. Taylor & Francis, 1968.

[56] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. Roy. Statist. Soc. Ser. B*, vol. 36, pp. 111–147, 1974.

[57] N. Shah, S. Balakrishnan, A. Guntuboyina, and M. Wainwright, "Stochastically transitive models for pairwise comparisons: Statistical and computational issues," *ArXiv e-prints*, 2015. [Online]. Available: http://adsabs.harvard.edu/abs/2015arXiv151005610S

[58] D. McMillen, *Quantile regression for spatial data*, ser. SpringerBriefs in Regional Science. Springer Berlin Heidelberg, 2012.

[59] R. Koenker, *Quantile regression*. Cambridge University Press, 2005.

[60] F. Gerber, *gapfill: Fill missing values in satellite data*, 2016, R package version 0.9.2.

[61] R Core Team, *R: A language and environment for statistical computing*, R foundation for statistical computing, Vienna, Austria, 2016. [Online]. Available: http://www.R-project.org/

[62] Revolution Analytics and S. Weston, *foreach: Foreach looping construct for R*, 2015, R package version 1.4.3. [Online]. Available: http://CRAN.R-project.org/package=foreach

[63] OpenMP architecture review board, "OpenMP application program interface, version 4.5," 2016. [Online]. Available: http://www.openmp.org

[64] MPI Forum, "Message Passing Interface (MPI) forum," 2016. [Online]. Available: http://www.mpi-forum.org

[65] C. Justice, J. Townshend, E. Vermote, E. Masuoka, R. Wolfe, N. Saleous, D. Roy, and J. Morisette, "An overview of MODIS land data processing and product status," *Remote Sens. Environ.*, vol. 83, no. 1–2, pp. 3–15, 2002, the Moderate Resolution Imaging Spectroradiometer (MODIS): a new generation of Land Surface Monitoring.

[66] A. Huete, K. Didan, T. Miura, E. Rodriguez, X. Gao, and L. Ferreira, "Overview of the radiometric and biophysical performance of the MODIS vegetation indices," *Remote Sens. Environ.*, vol. 83, no. 1–2, pp. 195–213, 2002, the moderate resolution imaging spectroradiometer (MODIS): A new generation of land surface monitoring.

[67] M. Mattiuzzi, *MODIS: MODIS acquisition and processing*, 2015, R package version 0.10-18. [Online]. Available: http://R-Forge.R-project.org/projects/modis/

[68] J. Dwyer and G. Schmidt, "The MODIS reprojection tool," in *Earth science satellite remote sensing*, J. Qu, W. Gao, M. Kafatos, R. Murphy, and V. Salomonson, Eds. Springer Berlin Heidelberg, 2006, pp. 162–177.

[69] R. J. Hijmans, *raster: Geographic data analysis and modeling*, 2015, R package version 2.3-0. [Online]. Available: http://CRAN.R-project.org/package=raster

[70] R. S. Bivand, E. Pebesma, and V. Gomez-Rubio, *Applied spatial data analysis with R, Second edition*. Springer, NY, 2013. [Online]. Available: http://www.asdar-book.org/

[71] E. J. Pebesma and R. S. Bivand, "Classes and methods for spatial data in R," *R News*, vol. 5, no. 2, pp. 9–13, 2005. [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

[72] D. Nychka, R. Furrer, and S. Sain, *fields: Tools for spatial data*, 2016, R package version 8.4-1. [Online]. Available: http://CRAN.R-project.org/package=fields

[73] D. Sarkar, *Lattice: Multivariate data visualization with R*. New York: Springer, 2008. [Online]. Available: http://lmdvr.r-forge.r-project.org

[74] H. Wickham, *ggplot2: Elegant graphics for data analysis*. Springer New York, 2009. [Online]. Available: http://had.co.nz/ggplot2/book

[75] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?—arguments against avoiding RMSE in

the literature," *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, 2014. [Online]. Available: http://www.geosci-model-dev.net/7/1247/2014/
[76] Python software foundation, *version 2.7*, http://www.python.org, 2016.
[77] GNU C Library, "Reference manual," 2016. [Online]. Available: http://www.gnu.org/software/libc/manual/
[78] MATLAB, "version 8.3.0 (r2014a)," Natick, Massachusetts, 2014.
[79] L. Eklundh and P. Jönsson, *TIMESAT 3.2 with parallel processing*, 2015, software Manual, Version 3.2. [Online]. Available: http://web.nateko.lu.se/timesat/docs/TIMESAT32_software_manual.pdf
[80] P. S. Beck, C. Atzberger, K. A. Hgda, B. Johansen, and A. K. Skidmore, "Improved monitoring of vegetation dynamics at very high latitudes: A new method using MODIS NDVI," *Remote Sens. Environ.*, vol. 100, no. 3, pp. 321–334, 2006.
[81] F. Wilcoxon, "Individual comparisons by ranking methods," vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968
[82] M. Sherman, *Spatial statistics and spatio-temporal data*. John Wiley & Sons, Ltd, 2010.

Florian Gerber received a B.Sc. degree in mathematics from the University of Bern (UNIBE), Switzerland, in 2010, and a M.Sc. degree in Biostatistics from the University of Zurich (UZH), Switzerland, in 2013. During his studies, he worked as a statistician at the Institute of Social and Preventive Medicine, UNIBE, and as an external statistical consultant for a pharmaceutical company. Since 2013, he is teaching assistant and PhD candidate in the group of Prof. Dr. Reinhard Furrer at the Institute of Mathematics, UZH.

Rogier de Jong received the M.Sc. degree in Earth science from Utrecht University, the Netherlands (NL) and his Ph.D. degree in environmental science from Wageningen University, NL. He joined the Remote Sensing Laboratories, University of Zurich, Switzerland, as research associate. Since 2015, he leads the group on Remote Sensing of Dynamic Vegetation Systems. His research interests include: remote sensing of vegetation, dynamics of terrestrial ecosystems, time series analysis, UAV and consumer-grade observation systems.

Michael E. Schaepman received the M.Sc. and Ph.D. degrees in geography from the University of Zurich (UZH), Zurich, Switzerland, in 1993 and 1998, respectively. In 1999, he was a Postdoctoral Researcher with the Optical Sciences Center, University of Arizona, Tucson, AZ, USA. In 2000, he was appointed as a Project Manager of the European Space Agency APEX Spectrometer. In 2003, he became the Full Chair of Geoinformation Science and Remote Sensing at Wageningen University, Wageningen, the Netherlands. In 2009, he was appointed as the Full Chair of Remote Sensing with UZH, where he is currently heading the Remote Sensing Laboratories, Department of Geography. He is the Director of the University Research Priority Program "Global Change and Biodiversity" and Dean of the Faculty of Science. His research interests include computational Earth sciences using remote sensing and physical models, with particular focus on the land-atmosphere interface using imaging spectroscopy.

Gabriela Schaepman-Strub Gabriela Schaepman-Strub received the M.Sc. and Ph.D. degree in geography from the University of Zurich (UZH), Switzerland, in 1999 and 2004, respectively. In 2001, she was a visiting scientist at Boston University, USA. In 2003, she started a postdoc at Wageningen University, the Netherlands. In 2009, she moved back to UZH as a group leader in spatial ecology and remote sensing. During the past 3 years, she chaired the CEOS Land Product Validation sub-group (LPV). Her interests are in land surface-atmosphere interactions using remote sensing and radiative transfer models, with particular focus on feedbacks of biodiversity changes to climate in the Arctic.

Reinhard Furrer was born in Switzerland on October 4, 1972. He received his Dipl. degree in mathematics from the Swiss Technical Institute of Technology in Lausanne, Switzerland, in 1998, and his Ph.D. degree in statistics from the same institution in 2002.

From 2002 to 2005 he was Postdoctoral fellow at the Geophysical Statistics Project, National Center for Atmospheric Research (NCAR), Boulder, CO, USA. From 2005 to 2009 he was Assistant professor at the Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO, USA. He is currently an associate professor in the Department of Mathematics and an affiliated faculty member of the Department for Computational Science, University of Zurich, Switzerland.

He is a member of the American Statistical Society, the Institute of Mathematical Statistics and the International Association of Mathematical Geology.

PAPER IV

Pitfalls in the Implementation of Bayesian Hierarchical Modeling of Areal Count Data: An Illustration Using BYM and Leroux Models

Florian Gerber & Reinhard Furrer

Paper published in the *Journal of Statistical Software*

doi:10.18637/jss.v063.c01

Journal of Statistical Software

January 2015, Volume 63, Code Snippet 1. http://www.jstatsoft.org/

Pitfalls in the Implementation of Bayesian Hierarchical Modeling of Areal Count Data: An Illustration Using BYM and Leroux Models

Florian Gerber
University of Zurich

Reinhard Furrer
University of Zurich

Abstract

Several different hierarchical Bayesian models can be used for the estimation of spatial risk patterns based on spatially aggregated count data. Typically, the resulting posterior distributions of the model parameters cannot be expressed in closed forms, and MCMC approaches are required for inference. However, implementations of hierarchical Bayesian models for such areal data are error-prone. Also, different implementation methods exist, and a surprisingly large variability may develop between the methods as well as between the different MCMC runs of one method. This paper has four main goals: (1) to present a point by point annotated code of two commonly used models for areal count data, namely the BYM and the Leroux models (2) to discuss technical variations in the implementation of a formula-driven sampler and to assess the variability in the posterior results from various alternative implementations (3) to give graphical tools to compare sample(r)s which complement existing convergence diagnostics and (4) to give various practical tips for implementing samplers.

Keywords: MCMC, GMRF, R, **openBUGS**, **geoBUGS**, **spam**, **INLA**, **CARBayes**.

1. Introduction

Maps of spatially aggregated count data are often noisy, making interpretation difficult. To overcome this problem, Bayesian hierarchical models (BHMs) are frequently used to identify a smooth pattern that may be explained using underlying covariates and spatial factors.

Depending on the precise problem, different types of BHMs may be adequate. A Poisson likelihood (data layer) is commonly used for count data. The second layer, often called the process layer, links the log risk to spatially structured and unstructured components as well as potential covariates. The remaining layer(s) contain(s) the priors. The so-called Besag-

York-Mollié (BYM) model (Besag, York, and Mollié 1991; Mollié 1996) is extensively used in the case of the areal count data of rare diseases. For an overview and comparison of the BYM and other models see Waller and Carlin (2010) and Lee (2011). Yet another alternative model is the so-called Leroux model (Leroux, Lei, and Breslow 1999). For this and more alternatives also see LeSage and Pace (2009).

There is no such thing as a free lunch: ease of interpretation has to be paid by complexity of implementation. Therefore, inference in such models requires carefully adapted Markov chain Monte Carlo (MCMC) approaches or elaborated integrated nested Laplace approximation (INLA) techniques. For MCMC simulations, the sampler can be built "by hand" or in a software environment, e.g., BUGS and its derivatives, may be used. Tailored samplers are often used in an academic exercise framework or for more complicated settings. In those cases, the specifications of the "full conditional" densities, and the acceptance probabilities need to be derived.

Due to model complexity and the multitude of tuning possibilities of the approaches, both solutions are error-prone. Further, many models are robust in the sense that that (slightly) incorrect implementations may still lead to reasonable estimates such that incorrect conclusions are difficult to avoid. This recently happened in the BYM model example in Furrer and Sain (2010), where the calculation of the acceptance probability of the sampler, and, thus the results were incorrect. Similarly, the MCMC sampler of the R package **INLA** (Rue, Martino, Lindgren, Simpson, and Riebler 2009b) exhibited several issues that had to be addressed by the maintainers.

This paper has four main goals: (1) to present a point by point annotated code of the BYM model implementation (in **openBUGS**, R and **INLA**) and the Leroux model (in R and **CARBayes**) (2) to discuss technical variations in the implementation of a formula-given sampler and assess the variability in the posterior results from the various implementations, (3) to give graphical tools to compare samplers or even to detect incorrect samplers which complement existing convergence diagnostics and (4) to give various practical tips for implementing samplers. As an aside, the paper should reassure novices that even though the implementation of a BHM is theoretically straightforward, the road may be steep or bumpy.

This paper is not about (1) comparing or ranking individual packages or software environments, (2) or about quantitative and formal testing procedures to compare random samples.

We have opted to illustrate the approaches with a "classical" dataset to avoid any unnecessary complications. More specifically, we choose to use the oral cavity cancer data, which is available in the R packages **spam** and **INLA**. The dataset consists of death counts y_i caused by oral cavity cancer for a 5-year period (1986–1990) in the $i = 1, \ldots, 544$ districts (*Landkreise und kreisfreie Städte*) of Germany (Knorr-Held and Best 2001; Rue and Held 2005). The expected number of cases e_i was derived using demographical data that allows us to display the standardized mortality ratios y_i/e_i (Figure 1, left map). Finally, the dataset comes with a matrix $\mathbf{A}$ that defines the neighborhood stucture of the districts. We use a notation similar to Rue and Held (2005) such that, for example, bold lower and upper case letters are used for vectors and matrices, respectively.

Sections 2 and 3 discuss the BYM and Leroux models by first introducing the model and then discussing the different software implementations. The sections are concluded by comparing the MCMC sample(r)s and by presenting further remarks. Finally, Section 4 points to software options for some extensions of the models and gives some additional hints for assessing MCMC

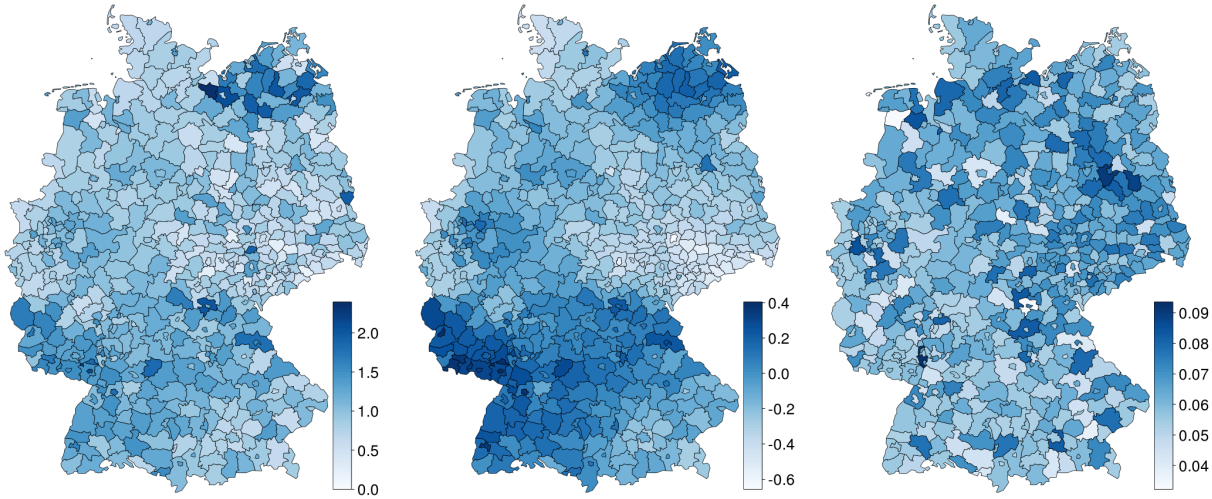


Figure 1: Standardized mortality ratios of oral cavity cancer deaths observed between 1986–1990 in Germany (left); the posterior means of the estimated relative log-risk of the BYM model (middle) and the difference between the posterior means of the log-risk of the Leroux and the BYM model (right).

sample(r)s. The source code to reproduce the MCMC chains, figures and tables is provided as supplementary material. In addition, the generated MCMC chains and the source code are available at http://www.math.uzh.ch/furrer/download/v63c01-code_with_data.zip.

2. Besag-York-Mollié model

In this section we give a short introduction to the BYM model, followed by implementations in **openBUGS**, R and **INLA**. These sections may be read independently of each other. The Sections 2.4 and 2.5 summarize the results and point to some extensions.

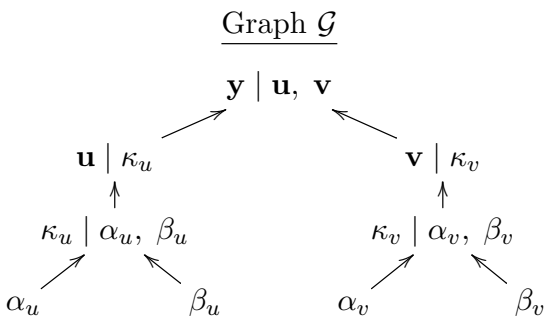
To explore the spatial distribution of the relative risk, the data y_i is assumed to be conditionally independent Poisson counts with rate $e_i \exp(\eta_i)$, yielding the likelihood

$$\pi(y_i \mid \eta_i) \propto \prod_{i=1}^{n} \exp\big(y_i\eta_i - e_i \exp(\eta_i)\big) = \exp\big(\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta})\big). \tag{1}$$

The log-relative risk is modeled by $\boldsymbol{\eta} = \mathbf{u} + \mathbf{v}$, where $\mathbf{v}$ is a zero mean white noise with precision κ_v, and $\mathbf{u}$ is a spatially structured component with precision κ_u. More precisely, $\mathbf{u}$ is a first order intrinsic Gaussian Markov random field (GMRF, Rue and Held 2005, Chaper 3)

$$\pi(\mathbf{u} \mid \kappa_u) \propto \kappa_u^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_u}{2}\sum_{i\sim j}(u_i - u_j)^2\Big) = \kappa_u^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_u}{2}\mathbf{u}^\top \mathbf{R}\mathbf{u}\Big),$$

where $i \sim j$ denotes the set of all unordered pairs of neighbors, i.e., regions sharing a common border, and hence the sum over all such sets can be written using a sparse "structure" matrix $\mathbf{R}$.



Level	Distribution
likelihood	Poisson
process model	IGMRF, *iid* Normal
prior	Gamma
hyper-prior	fixed

Figure 2: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

The resulting model is termed the *Besag-York-Mollié* model, see Besag *et al.* (1991). For inference on $\{\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v\}$ we set independent gamma priors for the precision parameters κ_u and κ_v, e.g., $\pi(\kappa_u \mid \alpha_u, \beta_u) \propto \kappa_u^{\alpha_u - 1} \exp(-\kappa_u \beta_u)$. We choose $\alpha_u = \alpha_v = 1$, $\beta_u = 0.5$ and $\beta_v = 0.01$ (and skip a carefully conducted sensitivity analysis, as this is not in the scope of this paper). This leads to the following posterior density

$$\begin{aligned}\pi(\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v \mid \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto\ & \kappa_v^{\alpha_v + \frac{n}{2} - 1} \kappa_u^{\alpha_u + \frac{n-1}{2} - 1} \\ & \times \exp\left\{-\kappa_v \beta_v - \kappa_u \beta_u + \mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta}) - \frac{\kappa_u}{2} \mathbf{u}^\top \mathbf{R} \mathbf{u} - \frac{\kappa_v}{2} \mathbf{v}^\top \mathbf{v}\right\}, \qquad (2)\end{aligned}$$

which is not a GMRF anymore. Since integration of this density is not feasible, we use MCMC sampling and approximation methods to estimate $\mathbf{u}, \mathbf{v}, \kappa_u$ and κ_v. In the reminder of this section we discuss different methods for this inference. Gibbs samplers implemented in **openBUGS**, two hand coded R implementations and the MCMC method of the R package **INLA**, as well as an integrated nested Laplace approximations from **INLA** are presented.

2.1. openBUGS implementation

The previously described model is a BHM with four levels. The dependency structure between the random variables (nodes), their levels, and distributions are shown in Figure 2.

Note that the graph $\mathcal{G}$ is directed and acyclic. Further, each node $\nu \in \mathcal{G}$ is independent of every other node given its parent nodes $pa(\nu)$. This implies that a factorization of the full joint distribution of all nodes in $\mathcal{G}$ is proportional to $\prod_{\nu \in \mathcal{G}} \pi(\nu \mid pa(\nu))$, which is used by the **openBUGS** engine to generate samples from the posterior distribution (Lunn, Jackson, Best, Thomas, and Spiegelhalter 2013).

The **openBUGS** language is used to specify such models and communicate them to the **openBUGS** engine. Similar to the graph representation, we define the distribution of each node given its parent nodes using an R like syntax. The model description is declarative, meaning that the order of the node-definitions is irrelevant. As usual, the symbol ~ stands for "is distributed as." In order to specify the spatially structured term $\mathbf{u}$, the `car.normal()` distribution from the **geoBUGS** extension is used. Since the distribution of $\mathbf{u}$ is implemented with a sum-to-zero constraint, we add an additional intercept with an improper flat prior. This construct is claimed to be equivalent to a spatially structured term $\mathbf{u}$ without constraint

(Lunn *et al.* 2013, p. 264). We save the following model to a text file 'model.txt'. Note that the code blocks corresponds to the levels likelihood, convolution-prior, and prior in Figure 2.

```
model{
  for(i in 1:N){
    Y[i] ~ dpois(landa[i])
    log(landa[i]) <- log(E[i]) + u[i] + v[i] }
  for(i in 1:N){ u[i] <- uConstr[i] + intercept }
  intercept ~ dflat()
  uConstr[1:N] ~ car.normal(adj[], weights[], num[], kappaU)
  for(k in 1:sumNumNeigh) { weights[k] <- 1 }
  for(i in 1:N){ v[i] ~ dnorm(0, kappaV) }
  kappaU ~ dgamma(1, 0.5)
  kappaV ~ dgamma(1, 0.01)
}
```

The observed and expected counts `Y` and `E`, as well as the neighborhood structure `adj` are saved in a separate text file 'data.txt' (see supplementary material). The arguments of the `car.normal()` distribution are: a sparse adjacency matrix `adj`, the number of regions connected in each row of the adjacency matrix `num[]` (also stored in 'data.txt'), the precision of `u`, i.e., `kappaU`, and a vector of 1's in `weight`. (We do not weight the adjacency structure). For another BUGS example of a BYM model see Bivand, Pebesma, and Gómez-Rubio (2013).

The R function `bugs()` from the R package **R2OpenBUGS** provides a convenient user interface to **openBUGS** (Sturtz, Ligges, and Gelman 2005). 300 000 samples from the posterior distribution are generated with the following call. A thinning of 20 and a burn-in of $5\,000 \times 20 = 100\,000$ is specified, resulting in 10 000 actually returned samples per variable:

```
R> library("R2OpenBUGS")
R> b <- bugs(model.file = "model.txt", data = "data.txt",
+    inits = function() { list(kappaU = 10, kappaV = 100, intercept = 1) },
+    parameters = c("kappaU", "u", "kappaV", "v"), n.iter = 15000,
+    n.burnin = 5000, n.thin = 20, n.chains = 1, bugs.seed = 2)
```

We manually set initial values for `kappaU` and `kappaV` via the argument `inits`. Further, the argument `parameters` specify variables for which samples are stored and returned to R. We only simulate one chain and set `n.chains = 1` for demonstration, but we recommend simulating several chains with different initial values that help to assess convergence. In fact, some comparisons in Section 2.4 are based on 200 different chains. **openBUGS** selects an appropriate sampling method for each node automatically.

Figure 3 shows diagnostic plots for the first 1 000 samples of the Markov chains for κ_u and κ_v, and Figure 1 shows the resulting posterior mean field of $\mathbf{u}$. We refer to Knorr-Held and Best (2001) for an (epidemiological) interpretation of the results.

2.2. Two R implementations

In the following, a hand coded R implementation (R Core Team 2014) of a Gibbs sampler with Metropolis-Hastings (MH) step is presented. To simplify notations of densities, the variables

$\mathbf{y}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ following the conditioning sign are omitted, e.g., we write $\pi(\mathbf{u}, \boldsymbol{\eta}, \kappa_u, \kappa_v)$ instead of $\pi(\mathbf{u}, \boldsymbol{\eta}, \kappa_u, \kappa_v \mid \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. We start the sampling procedure by first sampling κ_u and κ_v from

$$\pi(\kappa_u \mid \mathbf{u}, \boldsymbol{\eta}) \propto \kappa_u^{\alpha_u + \frac{n-1}{2} - 1} \exp\left\{-\kappa_u \beta_u - \frac{\kappa_u}{2} \mathbf{u}^\top \mathbf{R} \mathbf{u}\right\},$$
$$\pi(\kappa_v \mid \mathbf{u}, \boldsymbol{\eta}) \propto \kappa_v^{\alpha_v + \frac{n}{2} - 1} \exp\left\{-\kappa_v \beta_v - \frac{\kappa_v}{2} \mathbf{v}^\top \mathbf{v}\right\}.$$

In a second step, the parameter $\mathbf{u}$ and $\boldsymbol{\eta}$ are updated jointly using a MH step. To do this, we rewrite $\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v)$, which resulted from the posterior distribution in Equation 2. Recall that $\mathbf{v}$ can be expressed as $\boldsymbol{\eta} - \mathbf{u}$.

$$\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v) \propto \exp\left\{\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta}) - \frac{1}{2} \begin{pmatrix} \mathbf{u}^\top, \boldsymbol{\eta}^\top \end{pmatrix} \begin{pmatrix} \kappa_u \mathbf{R} + \kappa_v \mathbf{I} & -\kappa_v \mathbf{I} \\ -\kappa_v \mathbf{I} & \kappa_v \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\eta} \end{pmatrix}\right\} \quad (3)$$

The idea is to construct a proposal density for $\mathbf{u}$ and $\boldsymbol{\eta}$ that is a GMRF and thus easy to sample from. In addition, the proposal density should approximate the density in Equation 3 well to achieve a reasonable acceptance rate. We use the second-order Taylor expansion of $\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta})$ around $\tilde{\boldsymbol{\eta}}$, which is $\boldsymbol{\eta}^\top \mathbf{b}(\tilde{\boldsymbol{\eta}}) - \frac{1}{2} \boldsymbol{\eta}^\top \operatorname{diag}(\mathbf{c}(\tilde{\boldsymbol{\eta}})) \boldsymbol{\eta}$, with $\mathbf{c}(\tilde{\boldsymbol{\eta}}) = \mathbf{e} \exp(\tilde{\boldsymbol{\eta}})^\top$ and $\mathbf{b}(\tilde{\boldsymbol{\eta}}) = \mathbf{y} + (\tilde{\boldsymbol{\eta}} - \mathbf{1}) \mathbf{c}(\tilde{\boldsymbol{\eta}})^\top$. This leads to the normal proposal density

$$q(\mathbf{u}^\star, \boldsymbol{\eta}^\star, \tilde{\boldsymbol{\eta}} \mid \kappa_u, \kappa_v) \propto \exp\left\{-\frac{1}{2} \begin{pmatrix} \mathbf{u}^\top, \boldsymbol{\eta}^\top \end{pmatrix} \begin{pmatrix} \kappa_u \mathbf{R} + \kappa_v \mathbf{I} & -\kappa_v \mathbf{I} \\ -\kappa_v \mathbf{I} & \kappa_v \mathbf{I} + \operatorname{diag}(\mathbf{c}(\tilde{\boldsymbol{\eta}})) \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\eta} \end{pmatrix} + \mathbf{b}(\tilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta}\right\}.$$

The proposals $\mathbf{u}^\star$ and $\boldsymbol{\eta}^\star$ are then accepted with probability

$$\alpha = \min\left\{1, \frac{\pi(\mathbf{u}^\star, \boldsymbol{\eta}^\star \mid \kappa_u, \kappa_v)}{\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v)} \frac{q(\mathbf{u}, \boldsymbol{\eta}, \tilde{\boldsymbol{\eta}}^\star \mid \kappa_u, \kappa_v)}{q(\mathbf{u}^\star, \boldsymbol{\eta}^\star, \tilde{\boldsymbol{\eta}} \mid \kappa_u, \kappa_v)}\right\}. \quad (4)$$

As $\tilde{\boldsymbol{\eta}}$ we could take any value, but an optimized choice improves the approximation, and, hence increases the acceptance rate of the sampler. We set $\tilde{\boldsymbol{\eta}}$ to an approximation of the posterior mode. The latter is derived using two steps of the Newton-Raphson algorithm, which (under regularity conditions) converges to the posterior mode. To be more specific, we first set $\tilde{\boldsymbol{\eta}}$ to the current value of the chain and derive $\mathbf{b}(\tilde{\boldsymbol{\eta}})$ and $\mathbf{c}(\tilde{\boldsymbol{\eta}})$. Next, we set $\tilde{\boldsymbol{\eta}} = \mathbf{b}(\tilde{\boldsymbol{\eta}}) / \mathbf{c}(\tilde{\boldsymbol{\eta}})$ and repeat this procedure twice. The same procedure with $\boldsymbol{\eta}^\star$ as the starting point is applied to find $\mathbf{b}(\tilde{\boldsymbol{\eta}}^\star)$ and $\mathbf{c}(\tilde{\boldsymbol{\eta}}^\star)$. This leads to a suitable acceptance rate of about 48% (Roberts and Rosenthal 2001). See Rue and Held (2005) for more details and other options to increase the accuracy of the proposal density. Note that a third Newton-Raphson iteration for finding $\tilde{\boldsymbol{\eta}}$ and $\tilde{\boldsymbol{\eta}}^\star$ does not increase the acceptance rate. With a single Newton-Raphson iteration, the chain may not converge at all.

Next, we guide the reader through the R code of the Gibbs sampler. Note the use of the R package **spam**, which provides fast methods for sparse matrix algebra (Furrer 2014; Furrer and Sain 2010). The package contains the oral cancer data and the corresponding adjacency matrix, which we load first. `n = 544` is the number of districts.

```
R> library("spam")
R> data("Oral")
R> attach(Oral)
R> path <- system.file("demodata/germany.adjacency", package = "spam")
R> A <- adjacency.landkreis(path)
R> n <- dim(A)[1]
```

We set a seed value to initialize the random number generator and the number of desired samples (300 000). Further, we define the same hyper-parameters as in the **openBUGS** implementation.

```
R> set.seed(2)
R> hyperA <- c(1, 1)
R> hyperB <- c(0.5, 0.01)
R> totalg <- 300000
```

We build some variables to store the samples and set initial values

```
R> upost <- vpost <- array(0, c(totalg, n))
R> kpost <- array(NA, c(totalg, 2))
R> accept <- rep(NA, totalg)
R> upost[1, ] <- vpost[1, ] <- rep(0.001, 544)
R> kpost[1, ] <- c(10, 100)
```

Now we construct some quantities, which are (repetitively) used during the sampling.

```
R> eta <- upost[1, ] + vpost[1, ]
R> C <- exp(eta) * E
R> diagC <- diag.spam(c(rep(0, n), C))
R> b <- c(rep(0, n), Y + (eta - 1) * C)
R> Qu <- R <- precmat.IGMRFirreglat(A)
R> pad(Qu) <- c(2 * n, 2 * n)
R> Qv <- as.spam(rbind(cbind(diag(n), -diag(n)), cbind(-diag(n), diag(n))))
R> Q <- kpost[1, 1] * Qu + kpost[1, 2] * Qv + diagC
R> struct <- chol(Q, memory = list(nnzcolindices = 6467))
R> uRuHalf <- t(upost[1, ]) %*% (R %*% upost[1, ]) / 2
R> vvHalf <- t(vpost[1, ]) %*% vpost[1, ] / 2
R> postshape <- hyperA + c(n - 1, n) / 2
```

We start the loop of the Gibbs sampler and repeat the following steps: sample κ_u and κ_v (1st block), find an optimized $\tilde{\boldsymbol{\eta}}$ using two Newton-Raphson iterations (2nd block), draw $\mathbf{u}$ and $\boldsymbol{\eta}$ from the Taylor expansion around $\tilde{\boldsymbol{\eta}}$ (3rd block), find $\boldsymbol{\eta}^\star$ and calculate the log-acceptance probability $\log(\alpha)$ (4th block), accept or reject the draw and accordingly update the parameters (5th block). Note that '*' in the equations corresponds to '_' in the R code.

```
R> for (i in 2:totalg) {
+    kpost[i, ] <- rgamma(2, postshape, hyperB + c(uRuHalf, vvHalf))
+    etaTilde <- eta
+    for(index in 1:2) {
+      C <- E * exp(etaTilde)
+      diagC <- diag.spam(c(rep(0, n), C))
+      b <- c(rep(0, 544), Y + (etaTilde - 1) * C)
+      Q <- kpost[i, 1] * Qu + kpost[i, 2] * Qv + diagC
+      etaTilde <- c(solve.spam(Q, b, Rstruct = struct))[1:n + n]
+    }
```

```
+    C <- exp(etaTilde) * E; diagC <- diag.spam(c(rep(0, n), C))
+    b <- c(rep(0, n), Y + (etaTilde - 1) * C)
+    Q <- kpost[i, 1] * Qu + kpost[i, 2] * Qv + diagC
+    x_ <- c(rmvnorm.canonical(1, b, Q, Rstruct = struct))
+    upost[i, ] <- x_[1:n]
+    eta_ <- x_[1:n + n]
+    vpost[i, ] <- eta_ - upost[i, ]
+    uRuHalf_ <- t(upost[i, ]) %*% (R %*% upost[i, ]) / 2
+    vvHalf_ <- t(vpost[i, ]) %*% vpost[i, ] / 2
+    etaTilde_ <- eta_
+    for(index in 1:2) {
+      C_ <- E * exp(etaTilde_)
+      diagC_ <- diag.spam(c(rep(0, n), C_))
+      b_ <- c(rep(0, 544), Y + (etaTilde_ - 1) * C_)
+      Q_ <- kpost[i, 1] * Qu + kpost[i, 2] * Qv + diagC_
+      etaTilde_ <- c(solve.spam(Q_, b_, Rstruct = struct))[1:n + n]
+    }
+    C_ <- exp(etaTilde_) * E
+    diagC_ <- diag.spam(c(rep(0, n), C_))
+    b_ <- c(rep(0, n), Y + (etaTilde_ - 1) * C_)
+    Q_ <- kpost[i, 1] * Qu + kpost[i, 2] * Qv + diagC_
+    logPost_ <- sum(Y * eta_ - E * exp(eta_)) -
+      kpost[i, 1] * uRuHalf_ - kpost[i, 2] * vvHalf_
+    logPost  <- sum(Y * eta - E * exp(eta)) - kpost[i, 1] * uRuHalf -
+      kpost[i, 2] * vvHalf
+    logApproxX_ <- - kpost[i, 1] * uRuHalf_ - kpost[i, 2] * vvHalf_ -
+      sum(.5 * eta_^2 * C) + sum(b * eta_)
+    logApproxX  <- - kpost[i, 1] * uRuHalf  - kpost[i, 2] * vvHalf -
+      sum(.5 * eta^2 * C_) + sum(b_ * eta)
+    logAlpha <- min(0, logPost_ - logPost + logApproxX - logApproxX_)
+    if (log(runif(1)) < logAlpha) {
+      uRuHalf <- uRuHalf_
+      vvHalf <- vvHalf_
+      eta <- eta_
+      b <- b_
+      C <- C_
+      accept[i] <- 1
+    } else {
+      accept[i] <- 0
+      upost[i, ] <- upost[i - 1, ]
+      vpost[i, ] <- vpost[i - 1, ]}
+  }
```

Finally, we eliminate a burn-in of 100 000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10 000 (not shown). Diagnostic plots are shown in Figure 3.

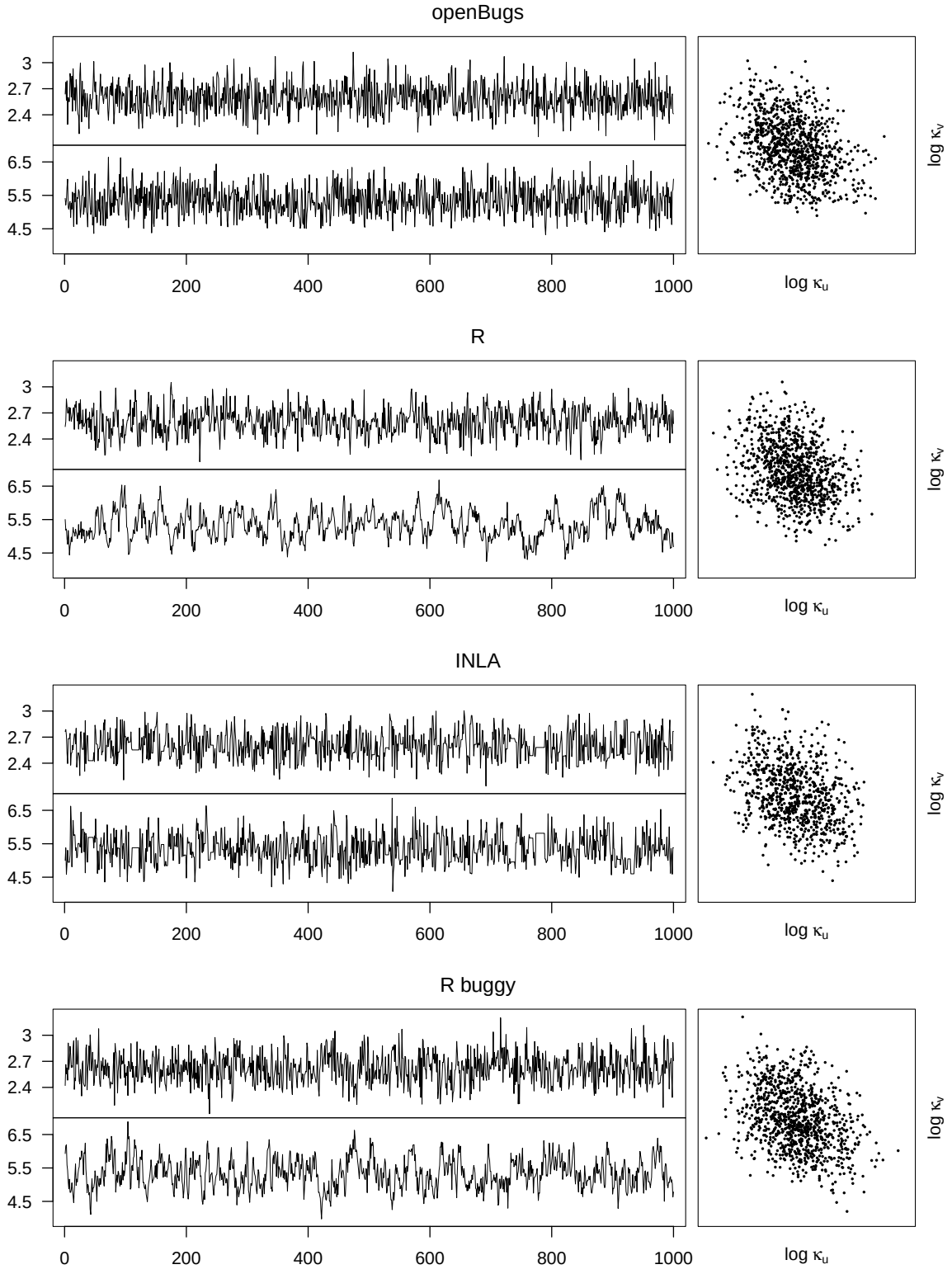


Figure 3: Diagnostic plots for the first 1 000 post burn-in samples from the **openBUGS**, R, **INLA** MCMC and R buggy BYM implementations are shown. Each panel consists of trace plots for $\log \kappa_u$ (upper) and $\log \kappa_v$ (lower), respectively. The mixing of $\log \kappa_u$ and $\log \kappa_v$ is shown in a scatter plot (right). The chains were already thinned with a factor of 20.

Incorrect R implementation

Furrer and Sain (2010) give an implementation of the BYM model that incorrectly calculates the acceptance probability. More specifically, on page 19, the line

```
R> factmp <- (postshape - 1) * (log(kstar) - log(kpost[ig - 1, 1]))
```

should read

```
R> factmp <- (postshape - 1) * (log(kstar) - log(kpost[ig - 1, ]))
```

Because only one element of `kpost[ig - 1, ]` is used (i.e., κ_v is set to κ_u), the acceptance probability is incorrect and overly high. Retrospectively, an acceptance rate of about 97% could have been an indication of an incorrectly calculated acceptance probability. The sampler uses the proposal density $q(\kappa_u^\star, \kappa_v^\star, \mathbf{u}^\star, \boldsymbol{\eta}^\star, \kappa_u, \kappa_v, \mathbf{u}, \boldsymbol{\eta})$ (jointly updating $\kappa_u, \kappa_v, \mathbf{u}, \boldsymbol{\eta}$), which is constructed on the previous value of $\boldsymbol{\eta}$ without using Newton-Raphson iterations. That leads to a very low acceptance rate, if the correct acceptance probability is used.

The sampler is available in its original version using `demo("article-jss-example2")` from **spam**. For this paper we have adjusted the burn-in, thinning, and sampling size parameters.

2.3. INLA implementation

As opposed to simulation based inference methods, the R package **INLA** uses nested Laplace approximations to estimate model parameters (Rue, Martino, and Chopin 2009a).

In order to fit the model, we first load the R package **INLA** and the oral cancer data. `path` contains the path to the corresponding adjacency matrix. The package, documentation and examples are available on http://www.r-inla.org/.

```
R> library("INLA")
R> data("Oral")
R> path <- system.file("demodata/germany.graph", package = "INLA")
```

Since **INLA** requires an index variable for each of the modeled components **u** and **v**, we have to duplicate the index column in the data frame.

```
R> Oral.inla <- cbind(Oral, region.struct = Oral$region)
```

Next, we define the model though a formula. The functions `f()` specify the priors for **u** and **v**, respectively. For **u** a regional structured prior is selected by setting `model = "besag"` and supplying a graph and a hyper-prior. We choose an unconstrained model (`constr = FALSE`), which implies that the intercept is absorbed by this random effect. Hence, the intercept is not identifiable, and we remove it from the formula through `-1` in the first line. The *iid* random effect **v** is specified in the second `f()` function. Note that `theta` corresponds to $(\log(\kappa_u), \log(\kappa_v))^\top$ and, therefore, '`loggamma`' priors are specified as having the same parameters as in the previous implementations.

```
R> formula <- Y ~ - 1 +
+   f(region.struct, model = "besag", graph = path,
```

```
+     hyper = list(theta = list(prior = "loggamma", param = c(1, 0.5))),
+     constr = FALSE) +
+   f(region, model = "iid",
+     hyper = list(theta = list(prior = "loggamma", param = c(1, 0.01))))
```

Internally, **INLA** reparametrizes by setting $\mathbf{x}^\top = (\mathbf{u}^\top, \boldsymbol{\eta}^\top)$, as is commonly done (Gelfand, Sahu, and Carlin 1995). Finally, we fit the model.

```
R> i.out <- inla(formula, family = "poisson", data = Oral.inla, E = E,
+    verbose = TRUE)
```

An alternative is to use the MCMC method of **INLA**, called by

```
R> wd.mcmc <- tempfile()
R> try(inla(formula, family = "poisson", data = Oral.inla, E = E,
+    working.directory = wd.mcmc, keep = TRUE,
+    inla.arg = "-m mcmc -N 300000 -T 20 -S .01", verbose = TRUE))
```

Currently, this MCMC method only works with the testing version of **INLA**. Additionally, the computations are carried out, but an error is returned to R. The results are nevertheless accessible in the temporary directory (path in `wd.mcmc`). We removed a burn-in of 100 000 samples and applied a thinning of 20. The results in this paper are based using the following **INLA** version:

```
R> inla.version()

INLA build date ...: Wed Feb 13 09:38:42 CET 2013
INLA hgid .........: hgid: 6d1015c52579  date: Wed Feb 13 09:28:06 2013 +0100
(output truncated)

R> sessionInfo()

R version 2.15.2 (2012-10-26)
Platform: i686-pc-linux-gnu (32-bit)
(output truncated)
```

The sampler of this particular implementation seems to work well. Diagnostic plots are shown in Figure 3. The acceptance rate was about 12%. However, the sampler in the current **INLA** testing version has a very low acceptance rate of about 3%. We hope that the issues will be addressed soon.

2.4. Comparison of the implementations

In order to keep the article at a reasonable length, we will not report convergence diagnostics of the individual samplers. Rather, we will focus on the comparison of the implementations. The assessment of "equality" of two samples is essentially the assessment, if two multivariate samples are drawn from the same distribution. Given the dimensionality of $\{\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v\}$, there is little hope that the formal tests presented in Rosenbaum (2005); Dhar, Chakraborty,

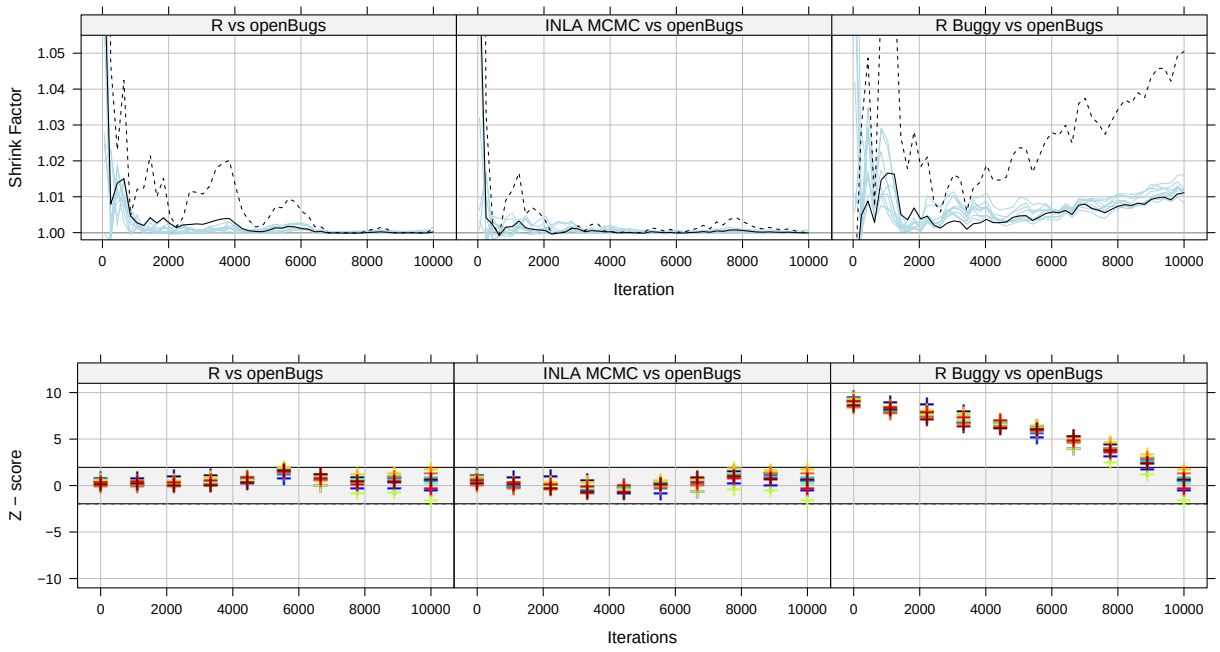


Figure 4: Comparison of the R, buggy R and **INLA** MCMC chains of κ_u against ten different κ_u chains from **openBUGS** using functions from R package **coda**. Upper panels: Gelman plots with the median shrink factor for 10 comparisons (sold lines) and the 95% quantile for the first comparison (dashed line). Lower panels: Geweke plots with one color per comparison. Here two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the `geweke.diag()` function to 0.5.

and Chaudhuri (2011) can be used. Alternatively, one can investigate individual posteriors (here κ_u, κ_v) and summary statistics of $\mathbf{u}$ and $\mathbf{v}$ (Wigley and Santer 1990; Li and Smerdon 2012).

We now illustrate a series of tools to compare two sample(r)s. Some of these are commonly used and reported here for completeness; others are new and complement the existing ones.

Figure 3 shows trace plots for $\log \kappa_u$ and $\log \kappa_v$ for the four different MCMC implementations. The trace plots of $\log \kappa_v$ for the R-implementations exhibit particularly high auto-correlations compared to those from **openBUGS**. This auto-correlation can be reduced by increasing the thinning value. An alternative is to jointly update all parameters of the models (κ_u, κ_v, $\mathbf{u}$, $\boldsymbol{\eta}$). See Knorr-Held and Rue (2002) for a discussion of different (block-) update procedures in a similar setting. The scatter plots of $\log \kappa_u$ and $\log \kappa_v$ in the same Figure show no suspicious pattern.

A numerical summary of the posterior distribution of κ_u and κ_v is given in Table 1, showing little evidence of issues with the incorrect R sampler. For the posterior mean, the naive standard error (SE) and the time-series standard error (TS SE) derived with the R package **coda** are given. With respect to the TS SE the variation in the mean estimates seems to be large.

In Figure 4, ten different chains for κ_u from **openBUGS** are compared against κ_u chains from the other sampler implementations. The diagnostic functions `gelman.plot()` and

	Implementation	Mean	SE	TS SE	SD	2.5%	50%	97.5%
κ_u	**openBUGS**	13.58	0.022	0.022	2.22	9.83	13.38	18.52
	R	13.62	0.022	0.035	2.22	9.91	13.37	18.57
	R buggy	13.93	0.023	0.029	2.29	10.13	13.72	19.01
	INLA	13.62	–	–	2.20	9.77	13.46	18.38
	INLA MCMC	13.62	0.022	0.025	2.21	9.91	13.41	18.52
κ_v	**openBUGS**	227.1	1.05	1.05	104.8	94.05	204.55	492.80
	R	231.5	1.10	3.51	110.0	93.47	207.57	513.59
	R buggy	231.6	1.11	2.60	111.0	94.27	204.91	516.22
	INLA	234.6	–	–	115.6	93.51	207.39	532.33
	INLA MCMC	231.3	1.10	1.44	109.9	94.06	206.53	516.35

Table 1: Summary table for the estimates of κ_u and κ_v generated with **openBUGS**, two hand coded R implementations, **INLA** and the MCMC-method of the **INLA** package. The estimates are calculated based on 10 000 samples of one chain (generated with 300 000 MCMC iterations in total). In addition to the standard error (SE), the time-series standard error (TS SE), derived with the R package **coda**, are given.

`geweke.diag()` from the R package **coda** are used (Plummer, Best, Cowles, and Vines 2006). In the Gelman plots (Brooks and Gelman 1998; Gelman and Rubin 1992), each line represents the median shrink factor of a comparison of two chains. For the first comparison (black line), the 95% quantile is drawn as dashed line. In the Geweke plot (Geweke 1992) 10 pairs of two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the corresponding R function to 0.5. Only the chains from the Rbuggy implementation seem to be different from the **openBUGS** ones. When we reduced the burn-in period from 100 000 to 5 000 samples, this effect was much less prominent.

A difficulty is often that the variability between individual chains is larger than between chains from different implementation methods. When using shorter chains this variability is even larger, and interpretation is more difficult. We recommend using empirical cumulative distribution functions (ECDFs) to compare realizations from different samplers (Figure 5, upper panels), although density estimates with the default choice of the kernel estimator and bandwidth selection of `density()` lead to an acceptable result too (Figure 5, lower left panel). When making these comparisons, we must keep in mind that the **INLA** approximations are tuned to be most accurate around the median. Hence, comparing ECDFs and tails of densities might lead to unfair comparisons. A functional boxplot approach (Sun, Genton, and Nychka 2012) may help to identify outlying densities. It calculates for each curve a (modified) band depth and orders the curves from the center outwards. The introduced measure defines functional quantiles and the outlyingness of a curve. The lower right panel of Figure 5 indicates that for κ_v all non-**openBUGS** runs are declared as outliers.

Figure 6 shows Q-Q-plots for the precision parameter κ_u. Empirical quantiles (from an arbitrary ordered sample) from the **openBUGS** sampler (x-axis) and the empirical quantiles from all others (y-axis) are drawn. For a better display, we have jittered the x-axis values while preserving the order. The quantiles of the incorrect sampler are shown as a red dashed line and are clearly set off from all the other lines.

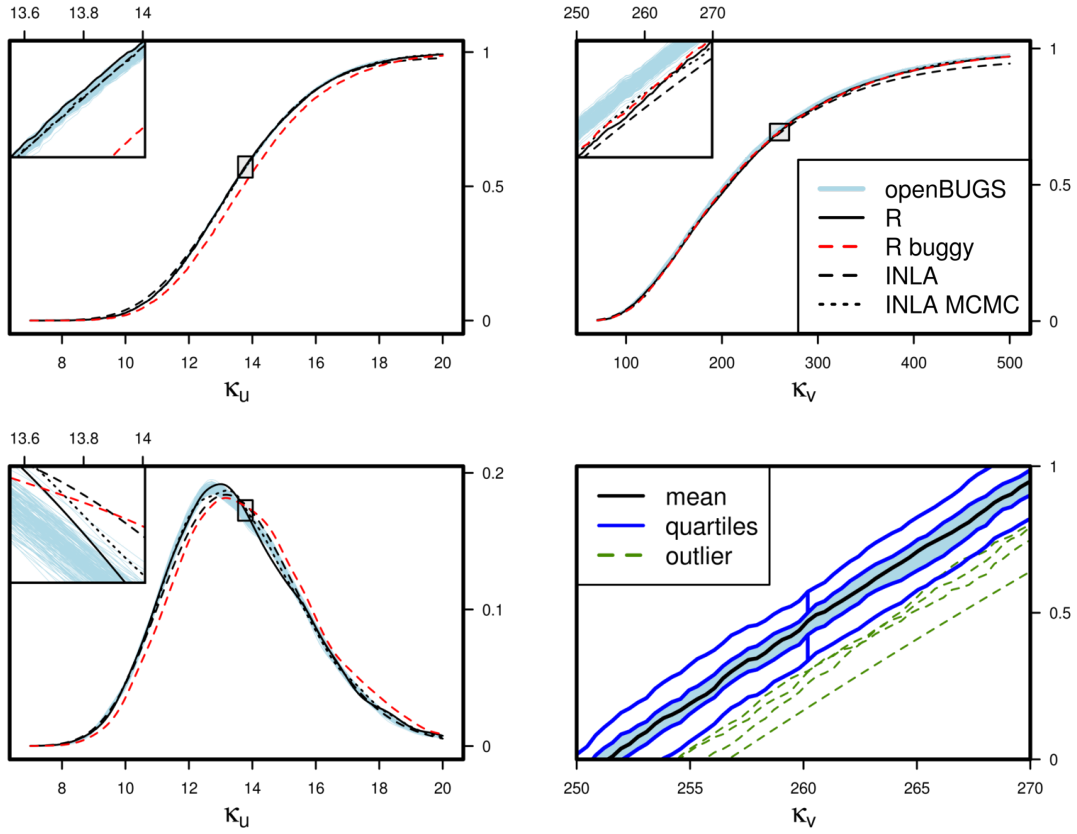


Figure 5: Comparison of the samples for κ_u (left) and κ_v (right) resulting from 200 **openBUGS** runs, the two R runs, and the **INLA** runs. Upper panels: empirical cumulative distribution functions (ECDFs). Bottom left panel: kernel density estimates with automatically chosen parameters (not recommended). Bottom right panel: functional boxplot where three **openBUGS**, the **INLA**, and the R buggy runs are marked as outliers (green lines).

Another alternative to compare sample(r)s is to use a simple clustering algorithm of the resulting empirical densities or distributions (Figure 7). Again, the dissimilarity (here measured by the classical Euclidean distance) is much larger between the incorrect sample and all the others. However, the **INLA** approximation stands out as well. Reducing the chain length and reducing the number of chains (or similarly increasing them) has little influence on the detection capability of the clustering approach.

Plots of posterior mean of the spatial fields **u** and **v** (like those shown in Figure 1) are very difficult to compare, as the differences are small and are masked by the spatial patterns. Other such "simple" diagnostic plots (e.g., median, standard deviation, IQR fields or differences of such) were not helpful to us either.

A more promising tool to compare spatial fields is the scatter plot, shown in Figure 8. There the spatial components averaged over all methods, except the buggy R, (x-axis) are plotted against those from the specific method on the y-axis. The mean absolute deviation D (also shown in the figure) is smaller then 1.61×10^{-3} for the **openBUGS**, R, and **INLA** implementations and 1.76×10^{-2} for the incorrect R-implementation. The corresponding values for **v** are 4.24×10^{-1} and 1.42×10^{-1}, respectively. Overall, the estimated **u** and **v** components of all implementations seem to agree well.

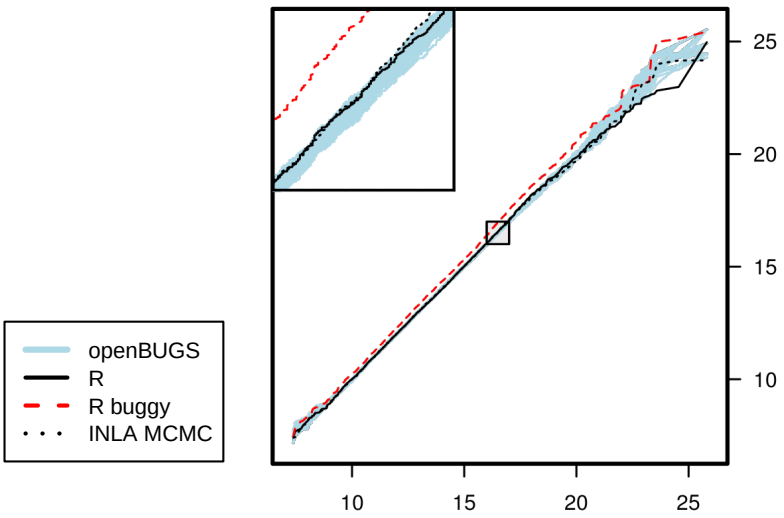


Figure 6: Q-Q-plots based on κ_u samples. The x-axis contains empirical quantiles (from an arbitrary ordered sample) from the **openBUGS** sampler, and the y-axis contains the empirical quantiles from all other **openBUGS** runs (blue lines), the R run (solid line), the incorrect (buggy) R run (red dashed line), and the **INLA** MCMC run (dotted line).

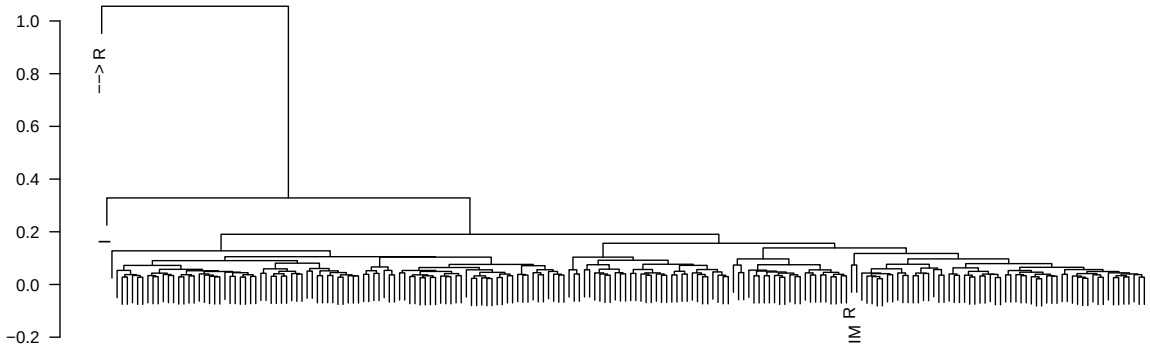


Figure 7: Dendrogram from a hierarchical clustering of the posterior ECDFs of κ_u of the 200 **openBUGS** runs (no symbol), the R (R), the buggy R (–> R), **INLA** (I) and the **INLA** MCMC (IM) runs.

A successful approach to discriminate sample(r)s and hence to identify an incorrect sampler is to take one method as a reference and plot the difference between the empirical densities or empirical distributions of reference and all the others (Figure 9). This is fast, and minor shifts or differences in scale are emphasized. A drawback is that the dependency structure of the samples and multiple testing issues are not taken into account. This makes the extension to a formal two-sample Kolmogorov-Smirnov test difficult.

Finally, a clustering algorithm detects again the wrong sampler by looking at (arbitrary) individual districts only. Here one chain is divided into ten subchains of a length of 1 000, and the empirical densities or distributions are calculated. For the district Sigmaringen, the

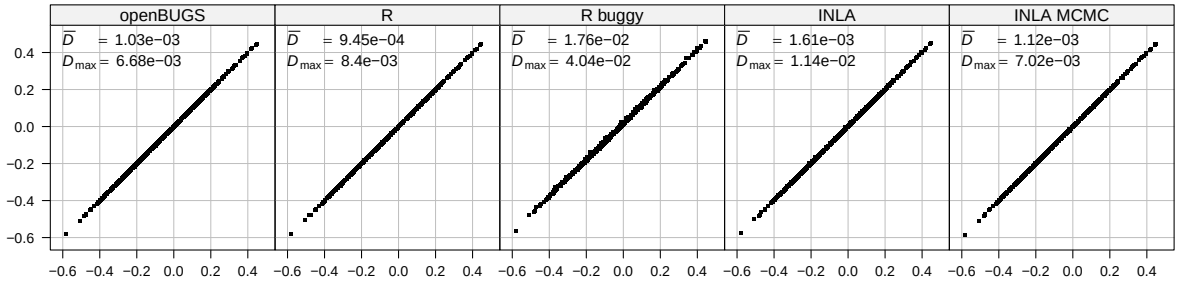


Figure 8: Comparison of the estimates for **u** of the different methods. x-axis: average mean estimates of the **openBUGS**, R and **INLA** implementations, y-axis: mean estimates of the specific method. $\overline{D}$ indicates the mean absolute deviation and $D_{\max}$ the maximum deviation.

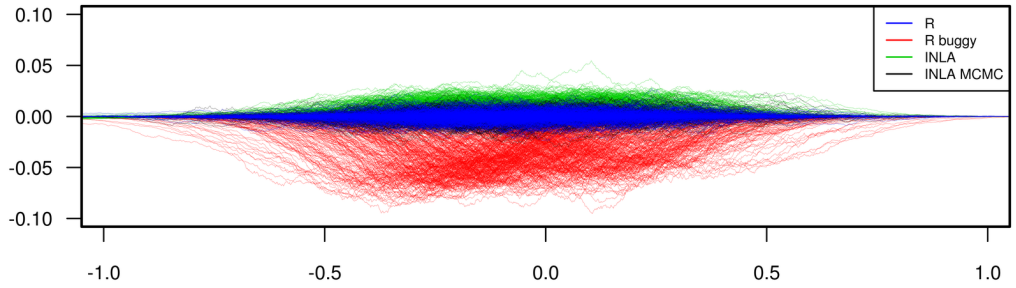


Figure 9: Comparison of posterior ECDFs for all 544 districts of the **u** component. One **openBUGS** run is used as a reference, and the differences to the R (blue), the incorrect R (red), and the **INLA** methods (green, black) are plotted.

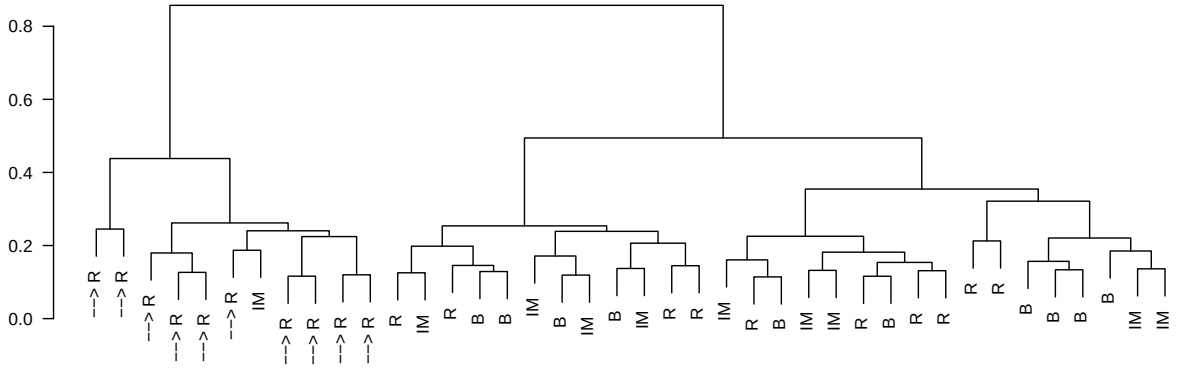


Figure 10: Dendrogram from a hierarchical clustering of the posterior ECDFs of the district Sigmaringen. Each chain was split into ten subchains. Chains from the **openBUGS**, the R, the incorrect R, and the **INLA** MCMC methods are denoted with 'B', 'R', '- -> R' and 'IM', respectively.

result of the clustering based on these 10×4 distributions is given in Figure 10. All subchains of the incorrect sampler are nicely grouped. An advantage of this clustering approach is that (dis)similarities of several chains of possibly different implementations are summarized in one step.

2.5. Discussion, extensions, pitfalls

The R implementation in Section 2.2 is the most flexible, but it is also the most demanding to code and is obviously the most error-prone. Because of the package **spam**, sparse matrix algebra can be done via a Fortran back-end, which leads to a reasonably fast Gibbs sampler featuring 30 iterations per second on an Intel 2GHz dual-core processor (approximately an hour of computation for a chain of a length of 300 000). Thus, R implementations are reasonably fast, as long as sparse matrices are used. Personal experience shows that, in all practical cases, the time spent to speed up the calculations does not offset the time gain (see Table 1 in Furrer and Sain 2010). Even more efficient ways may involve the concept of a just-in-time compiler for R code implemented in the R package **compiler**, which is part of the R base packages (R Core Team 2014). Another option is to run several chains in parallel using the R package **snowfall** (Knaus 2013). Finally, implementing (parts of) the model in C++ would reduce calculation time; for an example see Gerber (2013). The **openBUGS** implementation is flexible too, and if the required distributions are available, as in our case, it is much simpler to use. The sampler achieved 84 iterations per second (approximately half an hour of computation for a chain of length 300 000). In the R package **INLA** the specification of the model is very user-friendly, but a potential extension to a not-included setting is difficult. However, many cases are included and we foresee further extensions in the near future. The **INLA** method is very fast (less than 4 seconds) and thus is useful where estimation has to be fast. The MCMC method of the R package **INLA** achieved 34 iterations per second, and we have no doubt that a stable MCMC implementation will be available soon.

An unconstrained random field should be identical to a constrained random field with an intercept with uniform prior. In **INLA** a simple flag switches between both cases. In **openBUGS** a constrained version is implemented, and one has to add an intercept manually for the unconstrained case. While we did not observe issues with either implementation here, they are not always exactly identical, as is also discussed in the next section.

More complex equality constraints are easily implemented in **INLA** by specifying the option `extraconstr` in `f` and in R via `spam::rmvnorm.const()`, for example. Some of the sampling engines use a so-called centering-on-the-fly approach, and the implications on the equilibrium distribution are not clear to us (see also Schrödle, Held, Riebler, and Danuser 2011).

While generating many (200) long chains with 300 000 iterations each, **openBUGS** was not able to provide so many non-identical chains. Among the 200 chains, 7 were identical in our case. It seems that we have hit some sort of periodicity of the seed in **openBUGS**.

In the following, we point to possible extensions of the BYM model. Covariates that are observed for each region can be included, for example. For the oral cavity data, we could examine the effect of smoking and estimate an adjusted spatially structured component by setting $\boldsymbol{\eta} = \mathbf{u} + \mathbf{v} + \alpha\mathbf{s}$, where $\mathbf{s}$ contains observed smoking covariates of each region. This model is termed "ecological regression" and can be fitted in **openBUGS**; see (Lunn *et al.* 2013, p. 267) and (Bivand, Pebesma, and Gómez-Rubio 2008, p. 327) and in **INLA** (Schrödle and Held 2010). The latter paper also discusses the extension to spatio-temporal disease mapping. Further, it is possible to model two or more diseases jointly using a so-called shared component model (Held, Natário, Fenton, Rue, and Becker 2005; Rue and Held 2005), which is provided in **INLA** through the function `besag2` and can be implemented in **openBUGS** as well. **openBUGS** also provides a slightly different approach via the `mv.car()` distribution, which extends `car.normal()` in a natural way. MacNab (2010) discusses the similarities between

ecological regression and shared component modeling and proposes an ecological regression model that allows the researcher to account for measurement errors in the observed covariates.

3. Leroux model

An alternative model for areal count data was introduced by Leroux *et al.* (1999). In contrast to the BYM model, it has only one random effect component. Without including an intercept or additional covariates, this random effect simply models the log-relative risk $\boldsymbol{\eta}$. To be consistent with the implementations from Section 3.2, we set $\mathbf{u} = \boldsymbol{\eta}$. The separation of spatially structured and *iid* variance is controlled by an additional parameter λ. To be more specific, the same likelihood function as in Equation 1 is used, and $\mathbf{u}$ is modeled by the intrinsic GMRF

$$\pi(\mathbf{u} \mid \kappa, \lambda) \propto \det{}_G(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(-\frac{\kappa}{2}\mathbf{u}^\top \mathbf{Q}(\lambda)\mathbf{u}\Big).$$

Where $\kappa > 0$ is a precision parameter, and $\det_G$ denotes the generalized determinant (i.e., the product of all non-zero eigenvalues). The parameter $\lambda \in (0, 1)$ defines the degree of the spatial dependency through $\mathbf{Q}(\lambda) = (1-\lambda)\mathbf{I} + \lambda\mathbf{R}$. With appropriate (uninformative) priors for κ and λ, we get the posterior distribution

$$\pi(\mathbf{u}, \kappa, \lambda) \propto \kappa^{\frac{n}{2}-1} \det{}_G(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(\mathbf{y}^\top\mathbf{u} - \mathbf{e}^\top \exp(\mathbf{u}) - \frac{\kappa}{2}\mathbf{u}^\top\mathbf{Q}(\lambda)\mathbf{u}\Big). \tag{5}$$

In the next section, an R implementation of a Gibbs sampler for the Leroux model is presented. Further, three variations of this Gibbs sampler are discussed in Section 3.2, and a comparison and discussion of these variations follow in Sections 3.4 and 3.5.

3.1. R implementation

To estimate the parameters of the Leroux model, we implement a Gibbs sampler in R and sample from the posterior distribution. First, we sample $\mathbf{u}^\star$ from a normal proposal. To this end we use a second-order Taylor expansion around $\tilde{\mathbf{u}}$ of the term $\mathbf{y}^\top\mathbf{u} - \mathbf{e}^\top\exp(\mathbf{u})$ from the joint density (Equation 5), yielding the approximation $\mathbf{u}^\top\mathbf{b}(\tilde{\mathbf{u}}) - \frac{1}{2}\mathbf{u}^\top\operatorname{diag}(\mathbf{c}(\tilde{\mathbf{u}}))\mathbf{u}$ with $\mathbf{c}(\tilde{\mathbf{u}}) = \mathbf{e}\exp(\tilde{\mathbf{u}})^\top$ and $\mathbf{b}(\tilde{\mathbf{u}}) = \mathbf{y} + (\tilde{\mathbf{u}} - \mathbf{1})\mathbf{c}(\tilde{\mathbf{u}})^\top$. This leads to the normal proposal density

$$q(\mathbf{u}, \tilde{\mathbf{u}} \mid \kappa, \lambda) \propto \exp\Big(\mathbf{u}^\top\mathbf{b}(\tilde{\mathbf{u}}) - \frac{1}{2}\mathbf{u}^\top\big(\operatorname{diag}(\mathbf{c}(\tilde{\mathbf{u}})) + \kappa\mathbf{Q}(\lambda)\big)\mathbf{u}\Big).$$

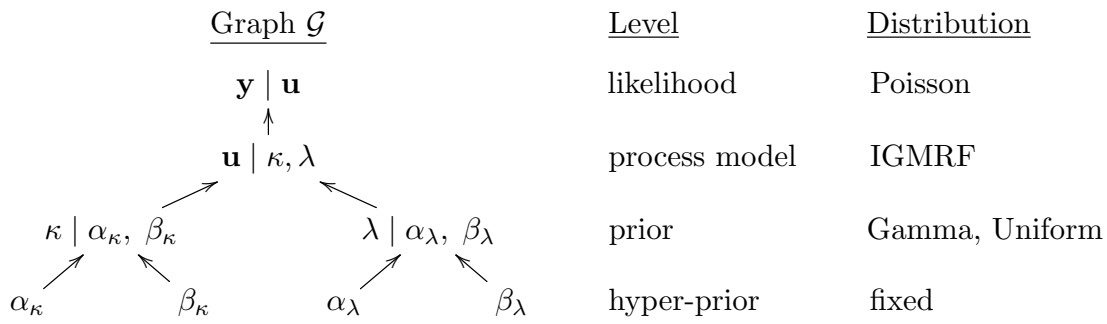


Figure 11: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

The proposal $\mathbf{u}^\star$ is accepted with probability $\min(1, \alpha_{\mathbf{u}})$, where

$$\log \alpha_{\mathbf{u}} = \log \left(\frac{\pi(\mathbf{u}^\star \mid \kappa, \lambda)}{\pi(\tilde{\mathbf{u}} \mid \kappa, \lambda)} \frac{q(\tilde{\mathbf{u}}, \mathbf{u}^\star \mid \kappa, \lambda)}{q(\mathbf{u}^\star, \tilde{\mathbf{u}} \mid \kappa, \lambda)} \right).$$

One Newton-Raphson iteration is applied to achieve an acceptance rate of $\approx 50\%$ (compare to Section 2.2). In a second step, κ is sampled from the full conditional

$$\pi(\kappa \mid \mathbf{u}, \lambda) = \kappa^{\frac{n}{2}-1} \exp\Big(-\frac{\kappa}{2} \mathbf{u}^\top \mathbf{Q}(\lambda) \mathbf{u} \Big).$$

Finally, λ is updated using a MH step again. This time we sample the proposal $\lambda^\star$ from a normal density truncated to $(0, 1)$, with the mean equal to the previous value of λ

$$q(\lambda^\star, \lambda \mid \mathbf{u}, \kappa) \propto 1_{(0,1)}(\lambda^\star) \exp\Big(-\frac{\tau}{2} (\lambda^\star - \lambda)^2 \Big).$$

The proposed $\lambda^\star$ is then accepted with probability $\min(1, \pi(\lambda^\star \mid \mathbf{u}, \kappa)/\pi(\lambda \mid \mathbf{u}, \kappa))$. The proposal density $q(\lambda^\star, \lambda \mid \mathbf{u}, \kappa)$ does not appear in the calculation of the acceptance rate, since it is symmetric in λ and $\lambda^\star$. $\tau > 0$ is a tuning parameter for the acceptance rate of λ.

Next, we show the R code for this version of the Gibbs sampler in more detail. First, the R packages **truncdist** (Novomestky and Nadarajah 2012), **spam** (Furrer 2014) and the Germany cancer data are loaded. The number of desired samples (300 000) and arrays for the posterior values are built. Note that we also initialize a `bpost` parameter, which we set to zero in each iteration. This is an artifact from other implementations mentioned in Section 3.2 and can be ignored.

```
R> library("spam")
R> library("truncdist")
R> data("Oral")
R> E <- Oral$E
R> Y <- Oral$Y
R> n <- 544
R> A <- as.matrix(adjacency.landkreis(
+    system.file("demodata/germany.adjacency", package = "spam")))
R> totaln <- 300000
R> upost <- array(NA, c(totaln, n))
R> bpost <- kpost <- lpost <- rep(NA, totaln)
R> accept <- array(0, c(totaln, 3), list(NULL, c("beta", "u", "lambda")))
```

Inital values are set in the first position of the corresponding posterior arrays.

```
R> bpost[1] <- 0
R> kpost[1] <- 15
R> lpost[1] <- 0.9
R> upost[1, ] <- rep(c(.1, -0.1), 544 / 2)
R> accept[1, ] <- 1
```

Next, a tuning parameter for the acceptance probability is set, and repeatedly used values are calculated.

```
R> lambda.proposal.sd <- 0.0408 * 1.74
R> R <- precmat.IGMRFirreglat(A)
R> eigenR <- eigen(R)
R> eigenR.value <- eigenR$values
R> Q <- (1 - lpost[1]) * diag.spam(544) + lpost[1] * R
R> Q.det <- sum(log(lpost[1] * eigenR.value + 1 - lpost[1]))
R> Q.struct <- chol.spam(Q)
R> postshape <- 0.5 * n - 1
```

We loop over the following steps of the Gibbs sampler: update β, which corresponds to setting it to zero (1st block), find an optimized $\tilde{\mathbf{u}}$ and update $\mathbf{u}$ with a MH step (2nd block), update κ (3rd block), and update λ with a MH step (4th block). Note that '$\star$' in the equations corresponds to '_' in the R code.

```
R> for (i in 2:totaln) {
+    bpost[i] <- 0
+    u.tilde <- upost[i - 1, ]
+    C <- E * exp(u.tilde)
+    B <- Y + (u.tilde - 1) * C
+    Q.tmp <- diag.spam(C) + kpost[i - 1] * Q
+    u.tilde <- c(solve.spam(Q.tmp, B))
+    C.tilde <- E * exp(u.tilde)
+    B.tilde <- Y + (u.tilde - 1) * C.tilde
+    Q.tilde <- diag.spam(C.tilde) + kpost[i - 1] * Q
+    u_ <- c(rmvnorm.canonical(1, B.tilde, Q.tilde, Rstruct = Q.struct))
+    u.tilde_ <- u_
+    C_ <- E * exp(u.tilde_)
+    B_ <- Y + (u.tilde_ - 1) * C_
+    Q.tmp_ <- diag.spam(C_) + kpost[i - 1] * Q
+    u.tilde_ <- c(solve.spam(Q.tmp_, B_))
+    C.tilde_ <- E * exp(u.tilde_)
+    B.tilde_ <- Y + (u.tilde_ - 1) * C.tilde_
+    log.alpha.u <- sum(Y * u_) - sum(E * exp(u_)) -
+      sum(Y * upost[i - 1, ]) + sum(E * exp(upost[i - 1, ])) +
+        sum(upost[i - 1, ] * B.tilde_) -
+          0.5 * t(upost[i - 1, ]) %*% (diag(C.tilde_) %*% upost[i - 1, ]) -
+            sum(u_* B.tilde) + 0.5 * t(u_) %*% (diag(C.tilde) %*% u_)
+    if(exp(log.alpha.u) > runif(1)) { upost[i, ] <- u_; accept[i, 2] <- 1 }
+    else { upost[i, ] <- upost[i - 1, ] }
+    kpost[i] <- rgamma(1, shape = postshape,
+      rate = 0.5 * upost[i, ] %*% (Q %*% upost[i, ]))
+    lambda_ <- rtrunc(n = 1, spec = "norm", a = 0, b = 1,
+      mean = lpost[i - 1], sd = lambda.proposal.sd)
+    Q_ <- (1 - lambda_) * diag.spam(544) + lambda_ * R
+    Q.det_ <- sum(log(lambda_ * eigenR.value + 1 - lambda_))
+    alpha.lambda <- exp(0.5 * (Q.det_ -
+      kpost[i] * upost[i, ] %*% (Q_ %*% upost[i, ]) - Q.det +
```

```
+       kpost[i] * upost[i, ] %*% (Q %*% upost[i, ]))
+     if(alpha.lambda > runif(1)) {
+       lpost[i] <- lambda_
+       Q <- Q_
+       Q.det <- Q.det_
+       accept[i, 3] <- 1
+     }
+     else {
+       lpost[i] <- lpost[i - 1]
+     }
+   }
```

Finally, we eliminate a burn-in of 100 000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10 000 (not shown). Diagnostic plots are shown in Figure 12. The acceptance probability is tuned to $\approx 49\%$ for the spatial parameter $\mathbf{u}$ and is $\approx 40\%$ for λ.

3.2. Three variations

As mentioned earlier, we implemented three additional variations of the Gibbs sampler. The variations differ in the way the random field is updated (1 block versus 55 updated blocks) and whether there is an intercept (with almost uninformative prior). The R code for the update of $\mathbf{u}$ in 55 blocks and for the update of the intercept was taken from **CARBayes** (Lee 2013). When possible, we just exchanged blocks of code to prevent errors. The corresponding R code is provided in the supplementary material of this paper. For obvious reasons, we call the sampler introduced in the last section '1 block, no intercept.'

1 block, intercept

Here $\boldsymbol{\eta}$ is modeled as $\beta + \mathbf{u}$ (i.e., an intercept β is added). For β an almost uninformative normally distributed prior with mean zero and variance 10^{10} is set. To guaranty identifiability of β and $\mathbf{u}$, a so-called centering-on-the-fly approach is implemented. It simply replaces $\mathbf{u}$ by $\mathbf{u} - \mathbf{1}^\top\mathbf{u}/n$ after each draw of $\mathbf{u}$ and might lead to some artifacts that are not entirely clear to us. Other options that simulate directly constrained $\mathbf{u}$'s are implemented, e.g., in **spam**. However, we decided to not use this in order to be consistent with the package **CARBayes**. The acceptance probability was tuned to $\approx 40\%$ for all parameters.

55 blocks, no intercept

Here the update of the spatial component $\mathbf{u}$ is separated into 55 blocks. This means that a proposal for the first block of size 10 is generated and accepted or rejected, then the second block is updated, and so forth. This version has no intercept and thus no sum-to-zero constraint. The acceptance probability was tuned to $\approx 40\%$ for all parameters.

55 blocks, intercept

In this variation the spatial component is updated in 55 blocks and an intercept β is added with an almost uninformative normally distributed prior (mean zero and variance 10^{10}). A sum-to-zero constraint on the spatial component is set. This variation corresponds almost

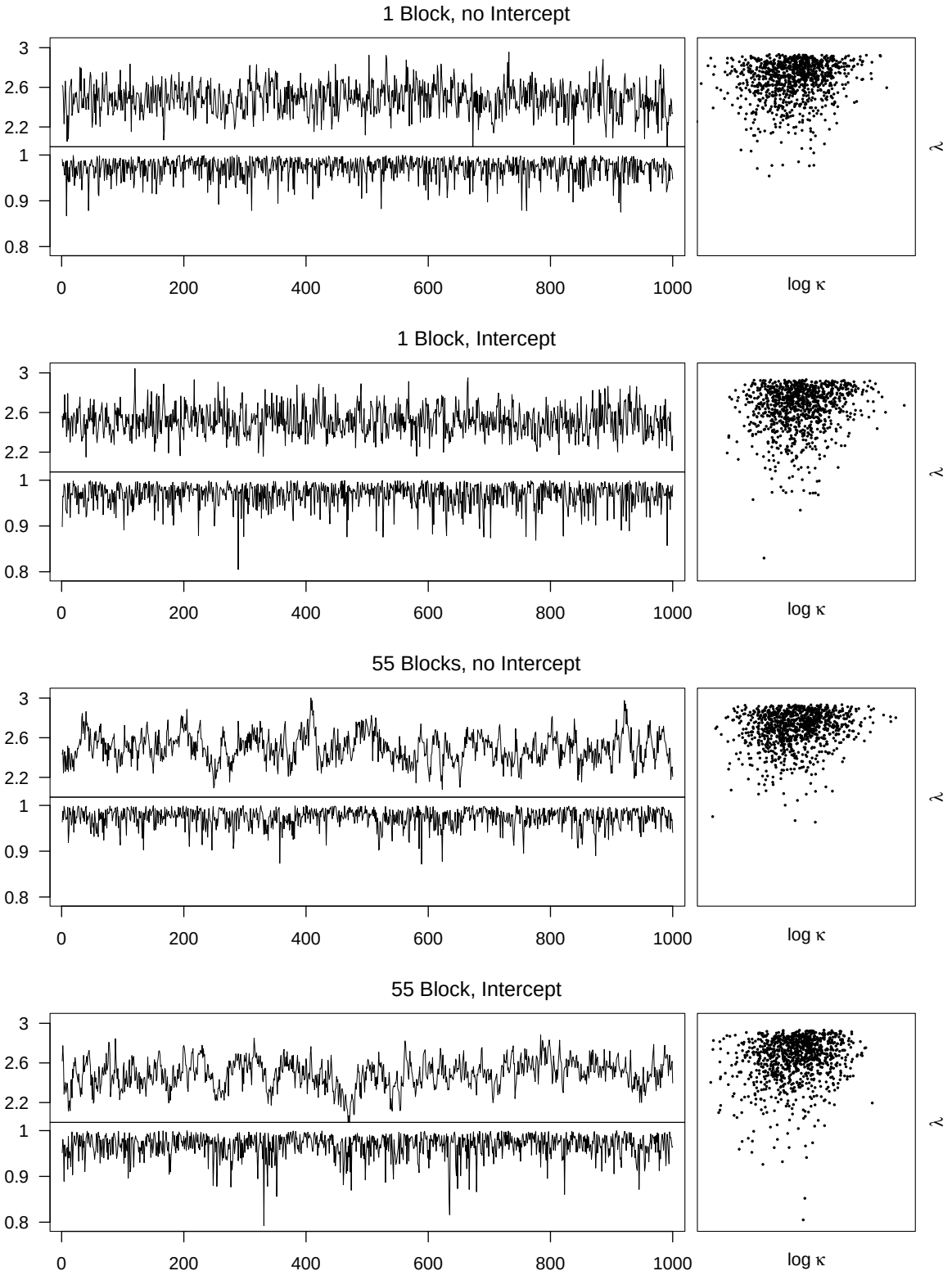


Figure 12: Diagnostic plots for the first 1 000 post burn-in samples from the four different Leroux implementations are shown. Each panel consists of trace plots for $\log \kappa$ (upper) and λ (lower), respectively. The mixing of $\log \kappa$ and λ is shown in a scatter plot (right). The chains were already thinned with a factor of 20.

	Implementation	Mean	SE	TS SE	MC SE	2.5%	50%	97.5%
κ	1 block, no intercept	12.13	0.0182	0.0229	0.0256	9.00	11.99	16.16
	1 block, intercept	12.47	0.0186	0.0243	0.0204	9.32	12.30	16.57
	55 blocks, no intercept	12.41	0.0184	0.0570	0.0643	9.22	12.26	16.46
	55 blocks, intercept	12.25	0.0179	0.0585	0.0636	9.18	12.12	16.08
λ	1 block, no intercept	0.972	0.00021	0.00022	0.00022	0.918	0.977	0.998
	1 block, intercept	0.969	0.00024	0.00025	0.00026	0.908	0.974	0.997
	55 blocks, no intercept	0.972	0.00021	0.00031	0.00030	0.921	0.977	0.998
	55 blocks, intercept	0.968	0.00024	0.00034	0.00034	0.909	0.973	0.997

Table 2: Summary table for the estimates of κ and λ generated with the four different implementations. The estimates are calculated based on 10 000 samples of one chain (generated with 300 000 MCMC iterations in total). Besides the standard error (SE), the time-series standard error (TS SE) derived with the R package **coda** and a Monte Carlo standard error (MC SE) based on 100 replications of the simulation are given.

to the one implemented R package **CARBayes**. The only difference is that the **CARBayes** implementation tunes the acceptance probabilities of the MH steps automatically. We run this sampler with 3 different configurations of the tuning parameters, yielding the acceptance probabilities $\approx 40\%$ for all parameters, $\approx 70\%$ for all parameters, and $\approx 35\%$ for the spatial, and $\approx 60\%$ for the other parameters, respectively. The latter is similar to the one resulting from the automatic tuning in the **CARBayes** implementation.

3.3. R package CARBayes

The R package **CARBayes** (Lee 2013) provides the function `poisson.leroux()`, which implements a Gibbs sampler similar to the version '55 blocks, intercept'. Additionally, this version allows us to specify explanatory variables using a formula interface. The function comes with a mechanism that tunes the acceptance probability automatically. We run the function with the same settings as our implementations.

```
R> library("CARBayes")
R> out <- poisson.lerouxCAR(formula = Y ~ offset(log(E)), data = Oral,
+    W = A, beta = 0, phi = rep(c(-0.1, -1), 544/2), tau2 = 1/15, rho = 0.9,
+    n.sample = 300000, prior.var.beta = 1e10, prior.max.tau2 = 1e10)
```

We eliminate a burn-in of 100 000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10 000 (not shown).

3.4. Comparison of the implementations

To compare the four different versions of the MCMC samplers from Section 3.1 and 3.2, we repeated the MCMC runs 100 times for all model implementations. Each chain had a length of 300 000 from which a burn-in of 100 000 was removed and a thinning of 20 was applied. Thus, we end up analyzing 100 chains of a length of 10 000 for λ, κ and $\mathbf{u}$ per implementation. The three additional versions (two with varying acceptance probability and one from the R package **CARBayes**) are only shown in Figure 14 and are mentioned in the discussion thereof.

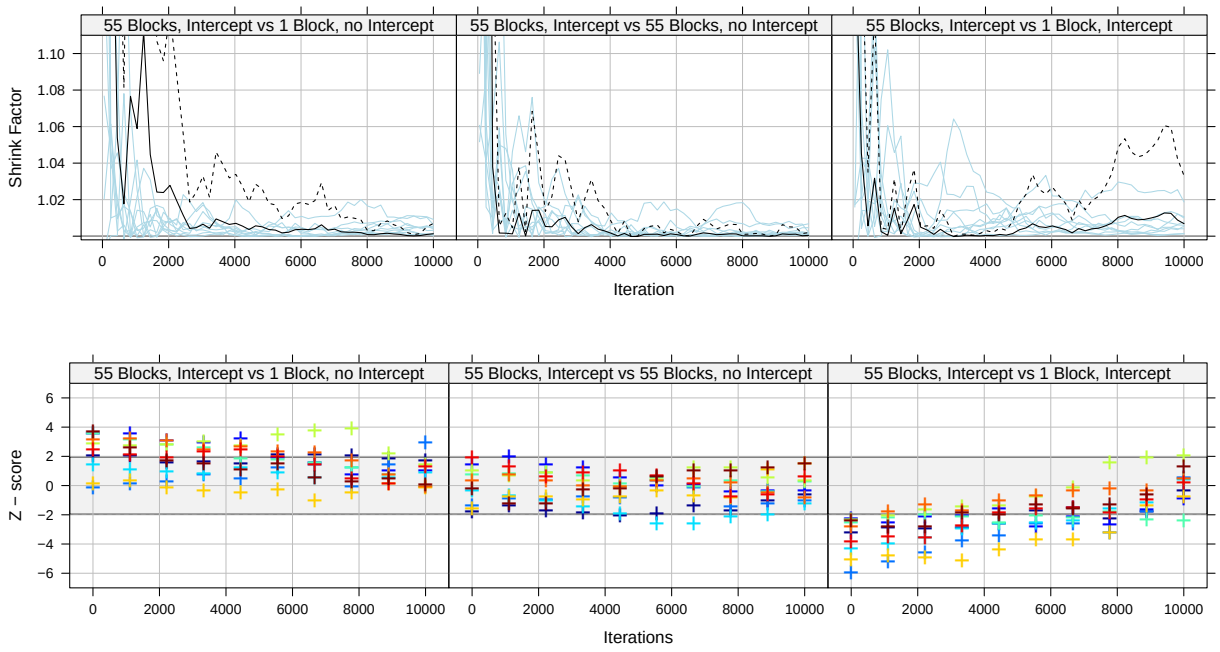


Figure 13: Comparison of 10 different κ chains from the variation '55 blocks, intercept' are compared against 10 κ chains from the other variations. The diagnostic functions are implemented in the R package **coda**. Upper panels: Gelman plots with the median shrink factor for 10 comparisons (solid lines) and the 95% quantile for the first comparison (dashed line). Lower panels: Geweke plots with one color for each comparison. The two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the `geweke.diag()` function to 0.5.

Again, many of the tools that we present are commonly used but are reported here for completeness. Figure 12 shows trace plots of $\log \kappa$ and λ for the four different MCMC variations. The trace plots of $\log \kappa$ of the implementations, with an update of the spatial component in 55 blocks, show higher auto correlations. This is consistent with the findings of Knorr-Held and Rue (2002) for the BYM model. The scatter plots of $\log \kappa$ and λ in the same figure show no suspicious patterns.

A numerical summary of the posterior distribution of κ and λ for the first of the 100 runs is given in Table 2. For the posterior mean the naive standard error (SE), the time-series standard error (TS SE) derived with the R package **coda** and a Monte Carlo standard error (MC SE) based on 100 replications of each simulation are given. With respect to the TE SE, the variation in the mean estimates seems to be large. Thus, for example, the posterior mean of λ for the implementation with an intercept are lower than those for the implementations without intercept.

In Figure 13, ten different κ chains from the variation '55 blocks, intercept' are compared against ten κ chains from the other sampler versions. The diagnostic functions `gelman.plot()` and `geweke.diag()` from R package **coda** were used. In the Gelman plots, each line represents the median shrink factor of a comparison of two chains. For the black line the 95% quantile is also drawn as dashed line. In the Geweke plot, ten pairs of two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the corresponding

R function to 0.5. Although, we analyze long chains, the variability within and between each implementation is striking.

An overview of the 100 mean estimates for κ and λ for all variations (including those with varying acceptance probability and **CARBayes**) is given in the upper panels of Figure 14. It is reassuring that the **CARBayes** version has a large overlap with the corresponding '55 blocks, intercept' version. The varying acceptance probability seems to have little effect. The estimates for κ from the variations '1 block, intercept' and '1 block, no intercept' have little overlap with the other versions and none with each other. For the λ estimates, again, the pattern of higher values for implementations without intercept are visible. A reason for this feature is that some variance of $\boldsymbol{\eta}$ is absorbed by the intercept and thus lacks the precision κ of the corresponding random effect $\mathbf{u}$. The same patterns are visible in Figure 14 (bottom), where the ECDF's of κ and λ for 400 chains are drawn. A separation of the versions with and without the intercept is clearly visible. Again the variation within each of the four implementations is considerable.

Finally, we applied a hierarchical clustering to the 100 ECDFs of the four variations. The results for the parameters κ, λ, and four randomly selected regions $\eta_1, \ldots, \eta_4$ are shown in Figure 15. For κ and λ, the same patterns (already mentioned above) are confirmed. For the η parameters no clear patterns resulted. This indicates that the spatial fields of all versions are similar, or more precisely, that the within implementation variance is larger than the between implementation variance. An exception is η_3 from the '1 block, intercept' version, which seems rather separated from the other versions.

3.5. Discussion, extensions, pitfalls

Naturally, many of the comments in Section 2.5 apply here as well, and we only mention a few relevant or new points. The simulation of 100 chains per version with a length of 300 000 each was only feasible within a reasonable amount of time due to a parallel implementation on a computer with several nodes. The R package **snowfall** (Knaus 2013) greatly simplified the simultaneous generation of several chains with different seed values for the random number generator. One chain took about one hour on a single processor for the '1 block' versions and about three hours for the versions with a '55 block' update of $\mathbf{u}$. This corresponds to 83 and 28 iterations per second, respectively. The difference in speed results mainly from the faster sparse matrix algebra Fortran back-end, which was accessed through the R package **spam** for the '1 block' versions. Some optimization of the R code with respect to speed would be possible. However, this would reduce the readability of the code. Implementing parts of the code in C++ would reduce calculation time and is planed for the **CARBayes** package (Lee 2013).

As was pointed out in the previous Section, the chains for the versions with intercept are different from the chains without (especially the λ chains). One possible explanation is that the normal prior of the intercept with variance 10^{10} was informative enough to lead to the higher λ values for versions without intercept. Another reason for this feature might be that some variance of $\boldsymbol{\eta}$ is absorbed by the intercept and thus reduces the strength of the unstructured component of the corresponding random effect $\mathbf{u}$. Finally, it could also be an artifact of the mean centering-on-the-fly approach, which possibly has an effect on the equilibrium distribution of $\mathbf{u}$. A way to overcome this issue is to use `rmvnorm.const()` from the R package **spam**.

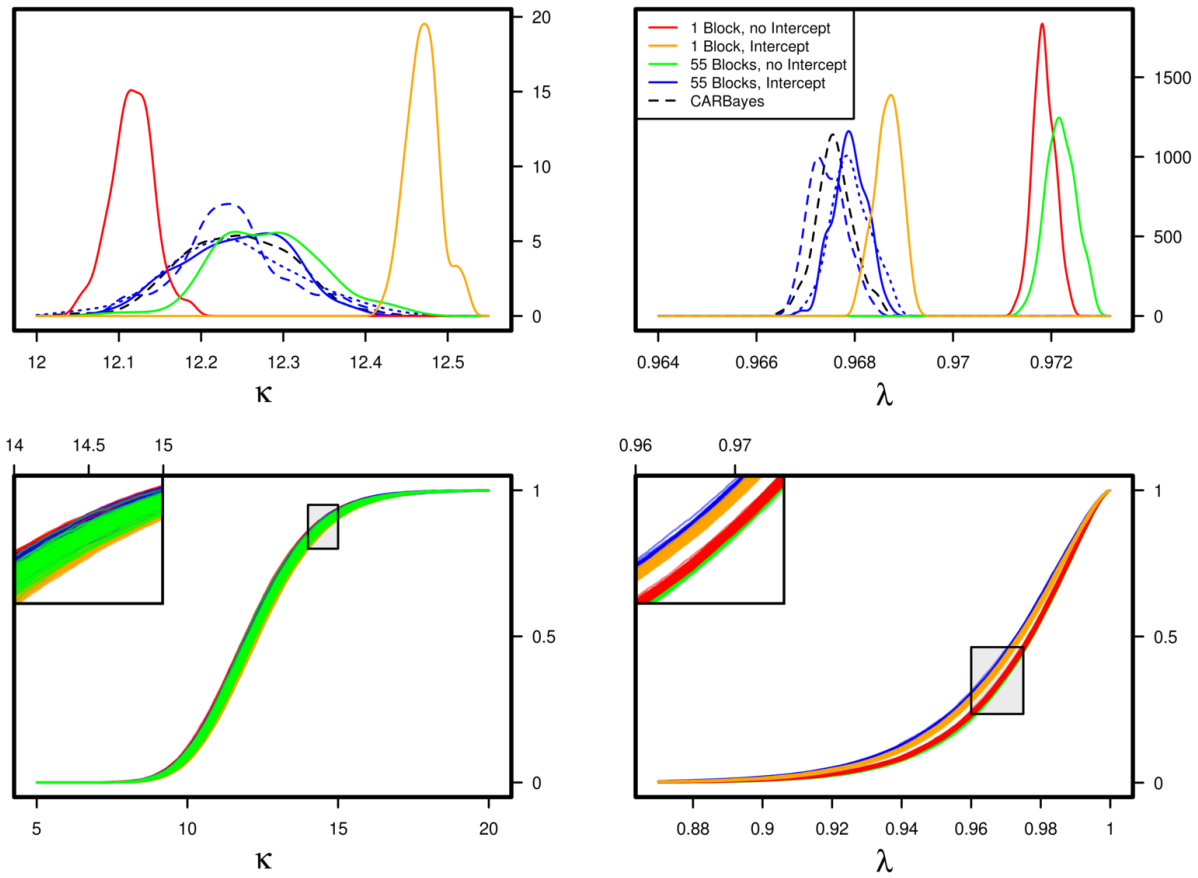


Figure 14: Upper panels: Densities of the mean estimates for κ and λ of 100 repeated simulations. The lines are either solid for an acceptance rate of $\approx 40\%$, dotted for an acceptance rate of $\approx 70\%$, or dashed for an acceptance rate similar to the **CARBayes** implementation. Lower panels: 100 ECDFs for κ and λ are drawn for the 4 different implementations with an acceptance rate of $\approx 40\%$. Each simulation considers 10 000 samples, which are taken from a MCMC run with 300 000 iterations.

To our knowledge there are no further implementations of the Leroux model readily available. Also, an extension of the model to the multivariate case with more than one diseases or a spatio-temporal model with a Leroux type of random effect would be interesting.

One possible extension of the model is "ecological regression", where additional covariates are taken into account. The R package **CARBayes** is capable of fitting such models, and a corresponding model description can be specified though the formula interface.

4. Final thoughts and remarks

Comparing several, long MCMC chains leads to the analysis of a huge number of data points. To summarize the information, several traditional methods, such as summary tables, Gelman plots, and Geweke plots, are useful (Tables 1, 2, Figures 4 and 13). In addition, we proposed a hierarchical clustering of ECDFs (Figure 7, 10 and 15). Our impression is that this approach

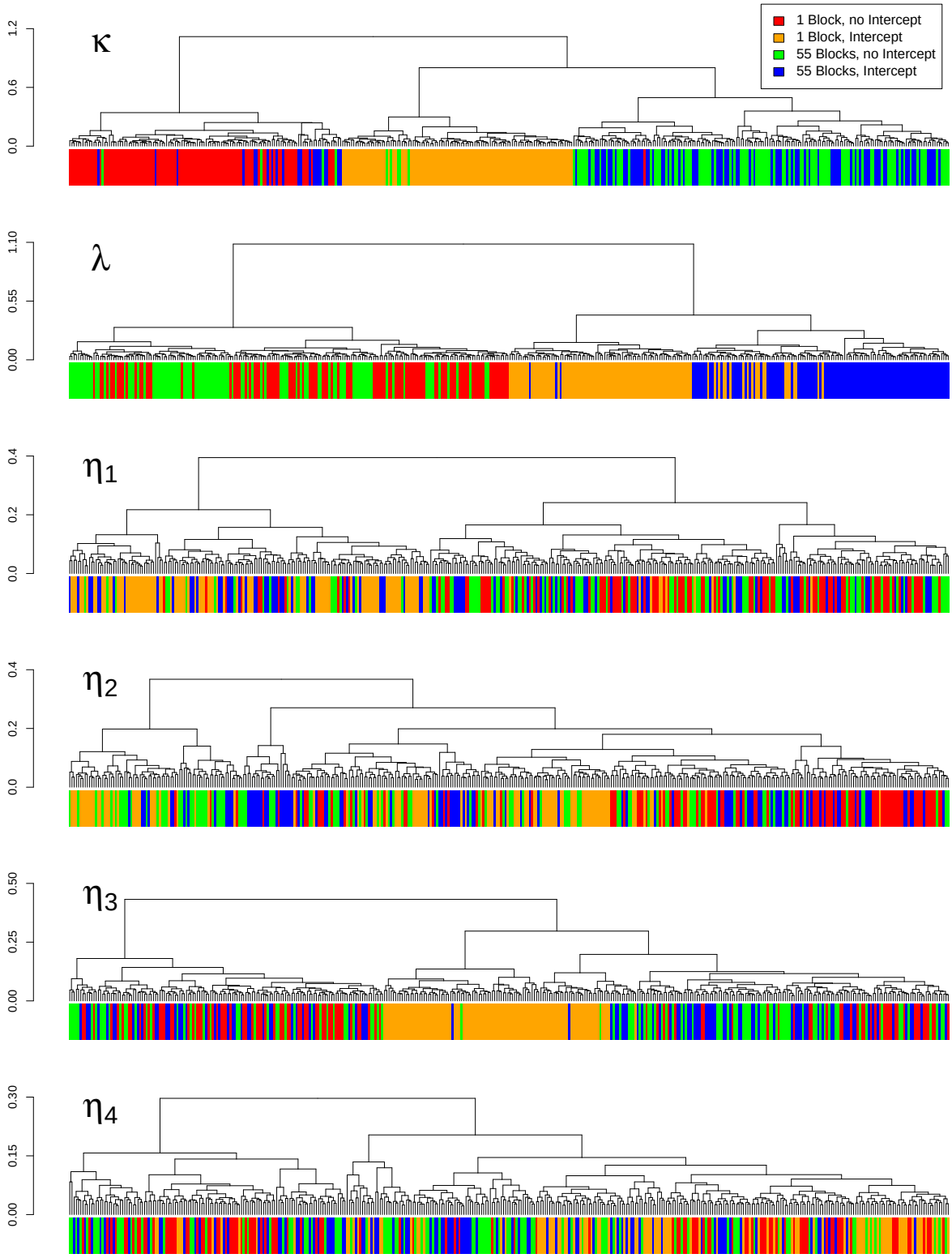


Figure 15: For the parameters κ, λ, and four randomly chosen elements of $\boldsymbol{\eta}$ dendrograms from a hierarchical clustering of the posterior ECDFs are shown. The clustering was applied to chains from the four implementation (indicated with colors) with 100 replications each. The figure is based on the same samples as Figure 14.

is useful to visualize the (dis)similarity of several implementations, especially if there are many replicates of chains available. Other approaches like principal component and canonical correlation analysis (Mardia, Kent, and Bibby 1979) did not lead to additional insights and are not shown in the paper. It would be interesting to compare the chains in a more formal ANOVA-like framework, which would lead to quantitative statements.

Our simulations suggest that the variability of the estimates derived from MCMC chains is considerable. One reason may be that the chains are too short (the analyzed chains had a length of 10 000 and were a sub-sample from an MCMC run with 300 000 iterations). It is the authors' impression that a decade ago it was "common" to run many chains of huge lengths ($> 10^6$) for relatively simple models (of few parameters). With rising computing power Bayesian models have become more and more complex, but the number of chains and chain lengths have not been increased; rather they have been severely decreased. It would be a very interesting bibliographic review to study the evolution of the number of parameters, the number of chains, and the chain lengths in articles published in statistical journals or in general scientific journals. This should be contrasted with the number of parameters of the BHMs and the available computing power.

Moreover, the variability between different implementations of the same model seems to be larger than the variability within one implementation and could even be systematic (compare to the λ parameter of the Leroux model for implementations with and without intercept, for instance). It would be interesting to find a formal explanation of these features in order to better understand the models and implementations. Further, an incorrectly calculated acceptance probability may lead to stable and unsuspicious chains, and it can be very difficult to detect such errors. But do such small differences matter in practice? We think that the following recommendations help identify differences that *do* matter: (1) many and long chains should be simulated, (2) if possible, different implementations should be used, (3) formal and graphical comparisons should be made.

Finally, the choice of the software environments used in this article was driven by our experiences. However, there exist more sampling engines, e.g., BayesX (Brezger, Kneib, and Lang 2005) or ADMB (Fournier, Skaug, Ancheta, Ianelli, Magnusson, Maunder, Nielsen, and Sibert 2012), with R interfaces **BayesX** (Kneib, Heinzl, Brezger, and Bové 2011) and **R2admb** (Bolker and Skaug 2012), or the package **geoRglm** (Christensen and Ribeiro 2002), that can handle BHMs with spatially correlated random effects. Different flavors of spatial models can also be handled with the JAGS engine (Plummer 2012) or with **spBayes** (Finley, Banerjee, and Carlin 2007; Finley, Banerjee, and Gelfand 2015). The community is extremely active, as indicated by the CRAN task view *Bayesian Inference* (Park 2014).

Acknowledgments

We thank the Editorial team, as well as the two reviewers for their constructive and valuable comments that led to a significant improvement in the paper. Further, we thank Håvard Rue for support on the **INLA** package. We acknowledge support of the UZH Research Priority Program (URPP) on "Global Change and Biodiversity" and the Swiss National Science Foundation (SNSF-143282 and SNSF-144973).

References

Besag J, York J, Mollié A (1991). "Bayesian Image Restoration, with Two Applications in Spatial Statistics." *The Annals of the Institute of Statistical Mathematics*, **43**(1), 1–20.

Bivand RS, Pebesma E, Gómez-Rubio V (2008). *Applied Spatial Data Analysis with R*. Springer-Verlag. URL http://www.asdar-book.org/.

Bivand RS, Pebesma E, Gómez-Rubio V (2013). *Applied Spatial Data Analysis with R*. 2nd edition. Springer-Verlag. URL http://www.asdar-book.org/.

Bolker B, Skaug H (2012). ***R2admb**: ADMB to R Interface Functions*. R package version 0.7.5.3, URL http://CRAN.R-project.org/package=R2admb.

Brezger A, Kneib T, Lang S (2005). "**BayesX**: Analyzing Bayesian Structural Additive Regression Models." *Journal of Statistical Software*, **14**(11), 1–22. URL http://www.jstatsoft.org/v14/i11/.

Brooks SP, Gelman A (1998). "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics*, **7**(4), 434–455.

Christensen OF, Ribeiro PJ (2002). "**geoRglm** – A Package for Generalised Linear Spatial Models." *R News*, **2**(2), 26–28. URL http://CRAN.R-project.org/doc/Rnews/.

Dhar SS, Chakraborty B, Chaudhuri P (2011). "Comparison of Multivariate Distributions Using Quantile-Quantile Plots and Related Tests." Unpublished manuscript, URL http://www.isical.ac.in/~statmath/html/publication/unblinded.pdf.

Finley AO, Banerjee S, Carlin BP (2007). "**spBayes**: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models." *Journal of Statistical Software*, **19**(4), 1–24. URL http://www.jstatsoft.org/v19/i04/.

Finley AO, Banerjee S, Gelfand AE (2015). "**spBayes** for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models." *Journal of Statistical Software*, **63**(13), 1–28. URL http://www.jstatsoft.org/v63/i13/.

Fournier DA, Skaug HJ, Ancheta J, Ianelli J, Magnusson A, Maunder MN, Nielsen A, Sibert J (2012). "AD Model Builder: Using Automatic Differentiation for Statistical Inference of Highly Parameterized Complex Nonlinear Models." *Optimization Methods and Software*, **27**, 233–249.

Furrer R (2014). ***spam**: Sparse Matrix*. R package version 1.0-1, URL http://CRAN.R-project.org/package=spam.

Furrer R, Sain SR (2010). "**spam**: A Sparse Matrix R Package with Emphasis on MCMC Methods for Gaussian Markov Random Fields." *Journal of Statistical Software*, **36**(10), 1–25. URL http://www.jstatsoft.org/v36/i10/.

Gelfand AE, Sahu S, Carlin B (1995). "Efficient Parametrization for Normal Linear Mixed Effects Models." *Biometrika*, **82**, 479–488.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**, 457–511.

Gerber F (2013). *Disease Mapping with the Besag-York-Mollié Model Applied to a Cancer and a Worm Infections Dataset*. Master's thesis, University of Zurich, Switzerland.

Geweke J (1992). "Evaluating the Accuracy of Sampling-Based Approaches to Calculating Posterior Moments." In JM Bernardo, J Berger, AP Dawid, JFM Smith (eds.), *Bayesian Statistics 4*, pp. 169–193. Oxford University Press, Oxford.

Held L, Natário I, Fenton SES, Rue H, Becker N (2005). "Towards Joint Disease Mapping." *Statistical Methods in Medical Research*, **14**(1), 61–82.

Knaus J (2013). ***snowfall**: Easier Cluster Computing (Based on **snow**)*. R package version 1.84-4, URL http://CRAN.R-project.org/package=snowfall.

Kneib T, Heinzl F, Brezger A, Bové DS (2011). ***BayesX**: R Utilities Accompanying the Software Package BayesX*. R package version 0.2-5, URL http://CRAN.R-project.org/package=BayesX.

Knorr-Held L, Best NG (2001). "A Shared Component Model for Detecting Joint and Selective Clustering of Two Diseases." *Journal of the Royal Statistical Society A*, **164**(1), 73–85.

Knorr-Held L, Rue H (2002). "On Block Updating in Markov Random Field Models for Disease Mapping." *Scandinavian Journal of Statistics*, **29**, 597–614.

Lee D (2011). "A Comparison of Conditional Autoregressive Models Used in Bayesian Disease Mapping." *Spatial and Spatio-Temporal Epidemiology*, **2**(2), 79–89.

Lee D (2013). "**CARBayes**: An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors." *Journal of Statistical Software*, **55**(13), 1–24. URL http://www.jstatsoft.org/v55/i13/.

Leroux BG, Lei X, Breslow N (1999). *Estimation of Disease Rates in Small Areas: A New Mixed Model for Spatial Dependence*. IMA Volumes in Mathematics and its Applications. US Government Printing Office.

LeSage J, Pace RK (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC.

Li B, Smerdon JE (2012). "Defining Spatial Comparison Metrics for Evaluation of Paleoclimatic Field Reconstructions of the Common Era." *Environmetrics*, **23**(5), 394–406.

Lunn D, Jackson C, Best N, Thomas A, Spiegelhalter D (2013). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Texts in Statistical Science. Chapman & Hall/CRC. URL http://www.mrc-bsu.cam.ac.uk/bugs/thebugsbook/.

MacNab YC (2010). "On Bayesian Shared Component Disease Mapping and Ecological Regression with Errors in Covariates." *Statistics in Medicine*, **29**(11), 1239–49.

Mardia KV, Kent JT, Bibby JM (1979). *Multivariate Analysis*. Academic Press, London.

Mollié A (1996). "Bayesian Mapping of Disease." In WR Gilks, S Richardson, DJ Spiegelhalter (eds.), *Markov Chain Monte Carlo in Practice*, pp. 359–379. Chapman & Hall, London.

Novomestky F, Nadarajah S (2012). ***truncdist**: Truncated Random Variables*. R package version 1.0-1, URL http://CRAN.R-project.org/package=truncdist.

Park JH (2014). *CRAN Task View: Bayesian Inference*. Version 2014-12-18, URL http://CRAN.R-project.org/view=Bayesian.

Plummer M (2012). ***rjags**: Bayesian Graphical Models Using MCMC*. R package version 3-7, URL http://CRAN.R-project.org/package=rjags.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Roberts GO, Rosenthal JS (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science*, **16**, 351–367.

Rosenbaum PR (2005). "An Exact Distribution-Free Test Comparing Two Multivariate Distributions Based on Adjacency." *Journal of the Royal Statistical Society B*, **67**(4), 515–530.

Rue H, Held L (2005). *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London.

Rue H, Martino S, Chopin N (2009a). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392.

Rue H, Martino S, Lindgren F, Simpson D, Riebler A (2009b). ***INLA**: Functions Which Allow to Perform Full Bayesian Analysis of Latent Gaussian Models Using Integrated Nested Laplace Approximaxion*. URL http://www.r-inla.org/.

Schrödle B, Held L (2010). "A Primer on Disease Mapping and Ecological Regression Using **INLA**." *Computational Statistics*, **26**(2), 241–258.

Schrödle B, Held L, Riebler A, Danuser J (2011). "Using Integrated Nested Laplace Approximations for the Evaluation of Veterinary Surveillance Data from Switzerland: A Case-Study." *Journal of the Royal Statistical Society C*, **60**(2), 261–279.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running **WinBUGS** from R." *Journal of Statistical Software*, **12**(3), 1–16. URL http://www.jstatsoft.org/v12/i03/.

Sun Y, Genton MG, Nychka D (2012). "Exact Fast Computation of Band Depth for Large Functional Datasets: How Quickly Can One Million Curves Be Ranked?" *Stat*, **1**(1), 68–74.

Waller L, Carlin B (2010). "Disease Mapping." In AE Gelfand, PJ Diggle, M Fuentes, P Guttorp (eds.), *Handbook of Spatial Statistics*. Taylor & Francis. Chapter 14.

Wigley TML, Santer BD (1990). "Statistical Comparison of Spatial Fields in Model Validation, Perturbation, and Predictability Experiments." *Journal of Geophysical Research*, **95**, 851–865.

Affiliation:

Reinhard Furrer, Florian Gerber
Institute of Mathematics
University of Zurich
8057 Zurich, Switzerland
E-mail: reinhard.furrer@math.uzh.ch, florian.gerber@math.uzh.ch
URL: http://www.math.uzh.ch/furrer/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 63, Code Snippet 1 *Submitted:* 2013-06-03
January 2015 *Accepted:* 2014-10-24

PAPER V

Challenges in linking Arctic plant biodiversity with satellite based landscape characterizations

Florian Gerber, Reinhard Furrer & Gabriela Schaepman-Strub

Technical report

Challenges in linking Arctic plant biodiversity with satellite based landscape characterizations

Florian Gerber[a], Reinhard Furrer[a,b], Gabriela Schaepman-Strub[c]

July 12, 2017

[a] Department of Mathematics, University of Zurich, Switzerland
[b] Department of Computational Science, University of Zurich, Switzerland
[c] Department of Evolutionary Biology and Environmental Studies, University of Zurich, Switzerland

Email addresses: florian.gerber@math.uzh.ch, reinhard.furrer@math.uzh.ch, gabriela.schaepman@ieu.uzh.ch

Keywords: International Tundra Experiment; Landsat; ASTER GDEM; Google Earth Engine; GLMM; lasso; biodiversity index; richness; evenness; Most Likely Transformations

Abstract

Climate warming triggers changes in the Arctic vegetation, which in turn feedback to global climate. Therefore, an improved quantification of the Arctic vegetation and changes thereof could lead to reduced uncertainties in climate predictions. However, quantifying vegetation from the available field measurements is challenging because they are very sparse in space and time. One approach to nevertheless obtain a complete description is to extrapolate (upscale) the field measurements with the help of remote sensing based landscape characterizations. While several statistical methods are suitable for that task, one necessary prerequisite for all upscaling methods is the existence of a correlation pattern between the field measurements and the remote sensing based landscape characterizations at locations where both data types are available. In the presented study we tried to establish such a correlation pattern for biodiversity field measurements. We used a synthesis of plant abundance field measurements available from the International Tundra EXperiment (ITEX), which we then transformed into species richness and evenness biodiversity indices. For the landscape characterization we used spectral Landsat 5 and 7 satellite data as well as the ASTER global digital elevation model (GDEM). Both types of satellite data were used to characterize the landscapes by extracting the mean and/or the standard deviation of values near locations with field measurements. As the radius of the used discs is not clear a priori, we examined different radii ranging from 100 m to 10 km. The results of our analyses suggest that the extracted summary statistics from the Landsat data and the ASTER elevation model do not correlate with the species richness and evenness indices derived from the ITEX data.

1 Introduction

Global climate change alters the environmental conditions of tundra vegetation. For instance, the average temperature in the Arctic has increasing by 2°C during the time period from 1979 to 2010 (IPCC, 2013, see graphs for the HadCRUT4 climate product, p. 880) and also cloud coverage, precipitation, snow, and sea ice coverage patterns have changed (Chapin et al., 2005; Serreze et al., 2000; Barnhart et al., 2016); see Hinzman et al. (2013) and ACIA (2005) for a more complete overview of changes in the tundra ecosystem. Studying how vegetation adapts to these changes is of relevance to better understand tundra ecosystems and their interactions with climate. Examples of ecosystem–climate interactions are permafrost thawing, which is expected to lead to a release of carbon stored in the soils (Zimov et al., 2006; Limpens et al., 2008; Blok et al., 2010) and shrub expansion, which changes the energy and water balances (Chapin et al., 2005; Blok et al., 2011; Myers-Smith et al., 2015).

This study is concerned with the quantification of biodiversity of tundra vegetation. Biodiversity is a concept to describe the spatial variability of biological organisms (Colwell, 2009). In order to derive biodiversity estimates from observations certain aspects are quantified with biodiversity indices. We focus on the *richness* biodiversity index defined as the number $S \in \mathbb{N}$ of distinct species in an area and the *evenness* index defined as $-\sum_{i=1}^{S} p_i \ln p_i / \ln S$, where $p_1, \ldots, p_S \in [0, 1]$ denote the proportions of individuals classified as species i (Hill, 1973). With those two indices both the number of species and their relative abundance can be summarized based on field measurements.

As collecting field measurements in the Arctic region is expensive, the available biodiversity estimates are very sparse in time and space. This makes it difficult to derive valid statements for larger regions of the tundra, and hence, limits their usage for studying climate relevant mechanisms. To obtain a more complete description of biodiversity the field measurements can be extrapolated with the help of remote sensing based landscape characterizations (Walker et al., 2003; Raynolds et al., 2008; Schaepman-Strub et al., 2013). Extrapolation methods require a relationship between the biodiversity field measurements and the remotely sensed observations at locations where both types of observations are available. We investigate the following three types of relationships:

First, we consider the *productivity–biodiversity* relation stating that plant productivity is related to biodiversity; see Huston (2014) and the references therein. Many studies found that this relationship follows a linear or a unimodal hump-shaped curve with large diversity at intermediate productivity (Mittelbach et al., 2001; Gillman and Wright, 2006). The productivity–biodiversity relation was also studied in the Arctic region (Virtanen et al., 2013). As the normalized difference vegetation index (NDVI) derived from optical satellite images is a proxy for plant productivity, the productivity–biodiversity relation could be exploited by relating such NDVI values to field biodiversity measurements (Jones and Vaughan, 2010). This motivates the following hypothesis:

H1: Remote sensing based NDVI values locally correlate with field biodiversity measurements.

Second, we consider the *landscape diversity–biodiversity* relationship. Stein et al. (2014) summarize qualitative support for that relation across different taxonomic groups and spatial scales. As main explanations for larger biodiversity in diverse landscapes they mention an increased amount of niches, more shelter from adverse environmental conditions, and an increased speciation frequency due to isolation. Tuanmu and Jetz (2015) proposed summary statistics to measure landscape heterogeneity from remote sensing based enhanced vegetation index (EVI), which they linked to bird species richness. Moreover, Jones et al. (2014) used an object based approaches to characterize tree diversity from Landsat data. This leads us to the following hypotheses:

H2: The spatial variability of values from optical satellite images and derived quantities such as the NDVI can be used to characterize the heterogeneity of landscapes. These quantities can be used to predict field biodiversity measurements.

Third, topographical information such as slope and aspect could be used to predict biodiversity. In particular, the slope determines the gravitational hydrological gradient and therefore influences the conditions for vegetation (Ostendorf and Reynolds, 1998). This motivates the hypothesis:

H3: Digital elevation models can be used to derive terrain information such as aspect and slope. This can be used to predict field biodiversity measurements.

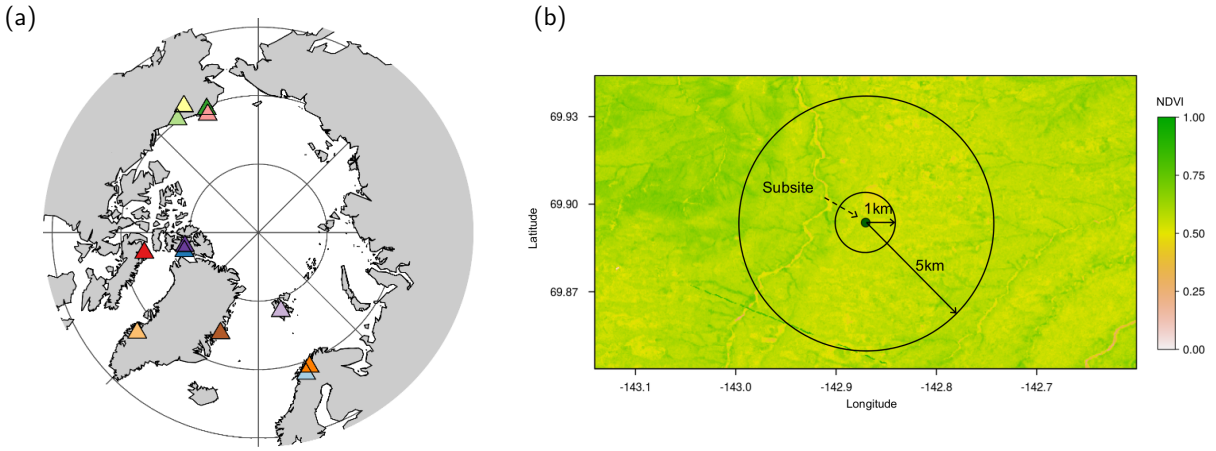


Figure 1: (a) geographical locations of the 12 considered study sites. Each triangle corresponds to one site. (b) illustration of how the satellite images were summarized into numbers. Shown is a NDVI maximum value composite (MVC) from Landsat covering a sub-site (green dot). For instance, the mean MVC NDVI value summarized at the radius of 5 km is defined as the mean of all values falling into the corresponding circle.

In the remainder of this manuscript we present our investigations of the hypotheses H1–H3. Section 2 introduces the datasets, statistical methods, and software used to generate the results presented in Section 3. The concluding discussion in Section 4 sheds light on potential issues with the data. In the Appendix (page 11) more information about the data and fitted models is given.

2 Method

2.1 Data

2.1.1 ITEX plant abundance data

In order to characterize vegetation compositions we considered field measurements available from the International Tundra EXperiment (ITEX, http://ibis.geog.ubc.ca/itex, Oberbauer et al., 2013). More precisely, the dataset CCIN10786_20120523 was used, which is a collection of field measurements from several research groups. The dataset was originally published in Elmendorf et al. (2012a,b) and features plot measurements from heated plots and control (untreated) plots. We only considered the plant abundance data from the control plots and used them to calculate the richness and species evenness biodiversity indices. The dataset features a hierarchical structure: multiple plot measurements are attributed to one sub-site and the sub-sites are group into sites. We restricted the analysis to the 12 sites above the polar circle ($66°33'$ N). A map of the geographical locations of the sites is given in Figure 1-a.

The spatial coordinates of the measurements were completely available at site level. However, sub-site coordinates were partially missing. By contacting the responsible investigators via email, we were able to get more spatial coordinates. We ended up with a complete set of sub-site coordinates for 7 of the 12 sites. For the remaining 5 sites, either the responsible person did not respond, the responsible person was not willing to share the coordinates, or the coordinates were not recorded. Tables 1 indicates the availability of spatial coordinates at sub-site level and the maximum distance between sub-site of a site ranging from 0.1 km to 76.4 km. To obtain a more homogeneous partition into site, we split the two largest sites ANWR and ABISKO into six and three sites, respectively. In addition to the spatial locations each measurement had between 2 and 11 replicates in distinct years.

After cleaning the datasets, we calculated the richness and the evenness biodiversity indices from the species counts at site and sub-site level for each available year. The resulting values are shown in Figure A1. As the temporal variability in the data exhibit no clear pattern and is small compared to the spatial variability, we aggregated the richness and the species evenness indices over time by taking the sub-site wise maximum and mean numbers, respectively. The resulting dataset has one richness and evenness value for each of the 72 sub-sites. For 53 (73%)

of sub-sites spatial coordinates were available. For the remaining 19 sub-sites we used the spatial coordinates of the site, and hence, some of these sub-sites have identical spatial coordinates and large uncertainties.

Table 1: Spatial information on the 12 considered test sites. "lat" and "long" are the average spatial coordinate. "no. sub-sites" indicates the number of sub-sites. "no. coordinates" indicates the number of distinct spatial coordinates per site. For the 5 highlighted sites spatial coordinates for some sub-sites were missing or identical. "max distance [km]" is the maximum distance between the sub-sites.

site	region	lat	lon	no. sub-sites	no. coordinates	max distance [km]
ANWR	Alaska	69.9	-144.1	27	27	76.4
ATQASUK	Alaska	70.5	-157.4	5	**1**	not applicable
BARROW	Alaska	71.3	-156.6	8	**1**	not applicable
TOOLIK	Alaska	68.6	-149.6	5	**3**	3.4
ALEXFIORD	Canada	78.9	-75.7	10	**6**	5.1
BYLOT	Canada	73.2	-80.0	2	2	4.1
SVERDRUP	Canada	79.1	-79.6	1	1	not applicable
KANGER	Geenland	67.1	-50.3	3	3	0.9
ZACKENBERG	Greenand	74.5	-20.5	2	2	0.1
ABISKO	Norway	68.3	18.7	3	3	22.2
KILPISJARVI	Norway	69.1	20.8	1	1	not applicable
SADVENT	Spizbergen	78.1	16.0	6	**3**	10.3

2.1.2 Landsat satellite imagery

In order to characterize landscapes Landsat Surface Reflectance products based on the Thematic Mapper (TM) carried on board the Landsat 5 satellite and the Enhanced Thematic Mapper (ETM+) carried on board the Landsat 7 satellite were used (Department of the Interior U.S. Geological Survey (USGS), 2017; Masek et al., 2006). Images from those products have a spatial resolution of $\approx 30\,\mathrm{m}$. We considered observations collected between January 1, 1984 and December 31, 2009. From Landsat 7 only observations collected before May 31, 2003 were used because of the systematic gaps caused by the Scan Line Corrector failure. Moreover, we relied on the provided quality assessment layer (based on the CFmask algorithm, Foga et al. 2017) to remove all contaminated values (i. e. , only values with `CFmask=0` were kept). We considered reflectance factors of the spectral bands B2 (green), B3 (red), B4 (near infrared), B5 (shortwave infrared 1), B7 (shortwave infrared 2), and, in addition, calculated an NDVI layer via

$$\mathrm{NDVI} = \frac{\rho_{\mathrm{B4}} - \rho_{\mathrm{B3}}}{\rho_{\mathrm{B4}} + \rho_{\mathrm{B3}}} \in [-1, 1], \tag{1}$$

where ρ_{B3} and ρ_{B4} are reflectance factors of the bands B3 and B4, respectively. Images covering the sub-site coordinates were sparse in time and partially contaminated by cloud cover; see Figure A2. To overcome this sparsity each layer was aggregated over time by calculating the pixel-wise maximum value with Google Earth Engine (Google Earth Engine Team, 2016); see Figure A3 for a screenshot of the Google Earth Engine online environment. As a result, the reflectance factors of each band are reduced to one maximum value composite (MVC) image covering all sub-sites. The described MVCs are based on values from the entire year. As it could be informative to consider only a shorter period of the growing phase of the vegetation, we additionally derived the MVC NDVI_JUNE, which is based on images collected in June only.

2.1.3 Digital elevation model from ASTER

The ASTER Global Digital Elevation Model (GDEM) version 2 was retrieved from the online tool "Reverb", http://www.echo.nasa.gov/reverb (NASA LP DAAC, METI, 2011). The data has a spatial resolution of the order of one arc second. We only considered "land values" by excluding pixels with an altitude of less than zero meters. From the GDEM aspect and slope layers were calculated taking into account eight neighboring pixels. The aspect layer took values between 0° and 360° and was transformed into a variable indicating the deviation from North taking values between 0° and 180° and a variable indicating whether the deviation is in western or southern direction.

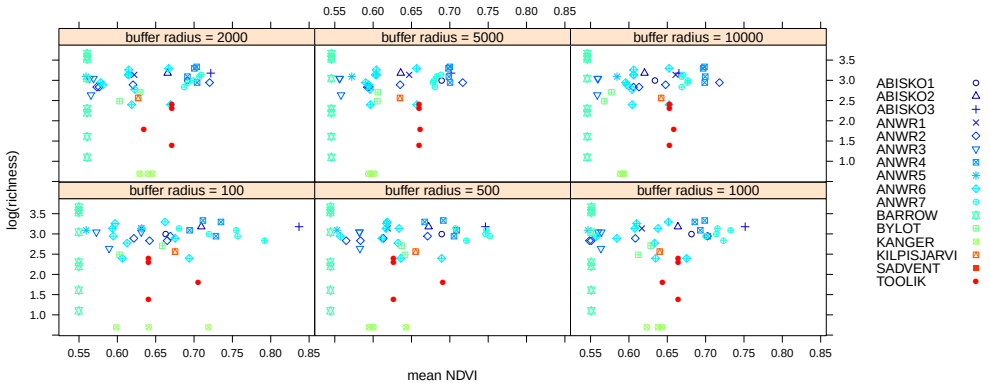


Figure 2: Scatter plot showing the means of the Landsat NDVI MVC (x-axis) and the corresponding log(richness) biodiversity index from the field measurements on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel. Similar figures for the other considered Landsat layers and summary statistics are given in Figures A6–A12.

2.2 Analyses

2.2.1 Characterizing landscape with remotely sensed data

The MVCs derived from Landsat (Section 2.1.2) and the slope and aspect layers from ASTER (Section 2.1.3) were used to characterize the sub-site locations. As it was not clear at which spatial resolution these layers contain relevant information, we summarized the images at multiple spatial resolutions. To that end, we defined buffer zones around the sub-site locations consisting of circles with radii of 100 m, 500 m, 1 km, 2 km, 5 km, and 10 km. Then, the values falling into the respective buffers are summarized by taking their mean value (Landsat and ASTER data) and their standard deviation (Landsat); see also Figure 1-b.

2.2.2 Statistical models

In all analyses, we considered linear predictors as statistically significant if their p-values were below 5%. However, it has to be noted that in the presented analyses many models were fitted. Hence, under to null hypothesis (no predictors are significant) it is expected that one out of 20 p-values is below 5% on average.

Visual inspection: To assess the relationships between the biodiversity indices (richness, evenness) at sub-site level and the summary statistics derived from the Landsat and ASTER GDEM based raster images, we first explored the data visually. More precisely, we created one scatter plot for each raster layer, each buffer radius, and each biodiversity index. The scatter plots show the values of the biodiversity index on the y-axis and the raster summary values on the x-axis; see Figure 2 for an example.

One model per satellite data layer: As a next step, we used a series of generalized linear mixed models (GLMMs) to verify whether the variability in the biodiversity data can be explained by the summary values from Landsat and ASTER GDEM (Bates et al., 2015); See also Appendix C for a brief introduction to mixed effects models. We fitted separate models for the two biodiversity indices richness and evenness. For the Landsat MVCs separate models were fitted for each raster layer and each buffer radius resulting in 42 models for richness and 42 models for evenness; the models are listed in Table A1 and A2, respectively. All those 84 models had the mean and the standard deviation of the considered layer at a given buffer size as linear predictors and a random intercept taking into account possible correlations of data within each site. The motivation to jointly model the mean and the standard deviation of the considered layers in one model is the fact that reflectance data have zero as a lower bound, which implies that the standard deviation is related to the mean value. Hence, considering the standard deviation without adjusting for the mean values would result in similar results as if only the mean values was considered. As richness consists of count data, we modeled it with a Poisson distribution and a log link function. Evenness was modeled with a

Gaussian distribution and the identity link function. For the slope and aspect layers from ASTER GDEM the models summarized in Table A3 and A4 were fitted. Models fits were summarized with coefficients of the linear predictors, p-values, and standardized effect sizes (Z-scores). The latter implies a significant result if the Z-score lies outside the interval $[-1.96, 1.96]$.

Joint analysis: The joint analysis of all considered Landsat layers and the ASTER GDEM data at a given buffer radius lead to a high dimension problem; i. e. , the number of predictors was large relative to the number of observations. To nevertheless fit a GLMM we used the lasso approach implemented in the R package glmmixedlasso to select relevant predictors (Schelldorfer et al., 2014).

Most Likely Transformations: In addition, we explored whether and how the Most Likely Transformations (MLT) framework can be used to predict the biodiversity indices with the help of the Landsat data and ASTER GDEM at new locations (Hothorn et al., 2015, 2014). The MLT approach provides a flexible and unified inference framework, which includes many classical regression models as spacial cases and goes beyond them (Hothorn, 2017). Our investigations were based on simulated data; see Appendix D for more information.

2.3 Software

All data manipulations and the model fitting was carried out with the statistical software R (version 3.2.5, R 2017) and Google Earth Engine (Google Earth Engine Team, 2016). The R packages raster, dplyr, and reshape2 simplified data handling; lme4, glmmixedlasso, and mlt were used to fit statistical models; ggplot2 and lattice were used for figure making.

3 Results

The construction of MVCs helped to overcome the sparsity of the Landsat data and to generate summary statistics for all layers at all sub-sites. Figure A4 and A5 show the distributions of days of the years (DOYs) of the used observations from the MVCs.

Visual inspection: To investigate the correlation structure between the biodiversity indices and the Landsat based landscape characterizations (see H1 and H2 of Section 1), their summary statistics (mean and standard deviation) for all radii and all layers were plotted against the richness and evenness indices. Figure 2 shows an example of such a plot and the complete set of figures is available in the appendix (Figures A6–A12). In these scatter plots no clear pattern of correlation between the biodiversity indices and the Landsat based landscape characterization could be detected.

One model per satellite data layer: The GLMMs of the Landsat data showed the two significant predictors highlighted in Table A1 and A2. A visualization of the significant predictors from those models is shown in Figure A13. As the number of significant predictors was low and did not exhibit a clear pattern, the models did not provide evidence against the null hypothesis being that the predictors from Landsat do not explain the variation in the diversity indices. The same conclusion followed when we considered the standardized effect sizes (Z-score) of all models of all Landsat layers as a function of the buffer radius (Figure A14).

In a next step we assessed the correlation structure between the biodiversity indices and the slope and aspect layers from ASTER GDEM (H3 of Section 1). It was possible to calculate summary statistics for all sites and buffer sizes. In Figure A15 summary statistics of the slope and aspect layers were plotted against the biodiversity indices. They do not show a correlation pattern. Moreover, we fitted GLMMs having summary statistics from the slope and aspect layers as linear predictors; the models are listed in Table A3 and A4, respectively. For the models with slope as linear predictors no predictor was significant; see Table 2. For the models with aspect as linear predictors 5 out of the 64 fitted models had a p-value below 0.05. However, the significant models showed a random pattern (Figure 3), which supports the hypothesis that the low p-values occurred because of the large amount of fitted models.

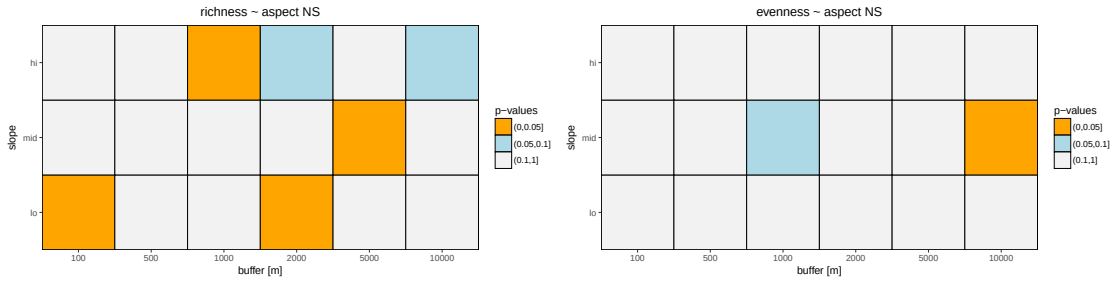


Figure 3: Pattern of significant predictors of the GLMMs having aspect as linear predictor; see also Table A4. No clear pattern can be detected.

Joint analysis: Eventually, we investigated whether it is possible to describe richness and evenness with models including all predictors from Landsat and ASTER GDEM calculated at a given buffer radius in one model. We tried to fit the models using the lasso generalized linear mixed models implemented in glmmixedlasso, but the models did not converge. This could indicate that not enough information was available to predict the biodiversity indices from the raster layers.

4 Discussion

The results presented above do not show evidence for the derived landscape characterizations being correlated with the field measurement based richness and evenness biodiversity indices. On the one hand, the reason for this could be the concept. This is, it may be impossible to relate biodiversity with the described of landscape characterizations. On the other hand, we faced a series of challenges during data preparation, which could have obscured correlations. The latter are discussed in the remainder of this section:

Plant abundance data from ITEX

- For 5 out of the 12 sites the geographical coordinates of the sub-sites were incomplete (Table 1). This made it impossible to match those sub-sites with the correct landscape characterization and we used the less precise site coordinates instead. The analysis could be improved if the spatial locations for all sub-sites or plot measurements would be available. In addition, it would be helpful to have more information about the accuracy of the indicated spatial coordinates, the sampled area, and the study design.
- The data are a synthesis of data collected by different research groups, which used several data collection methods. This may make it impossible to analyze the data jointly without loosing much information. For

Table 2: Results of the (generalized) linear mixed models having the sub-site species richness or species evenness as response variable and the mean slope from ASTER GDEM extracted at the indicated buffer size as response variables. We considered $\log(slope)$ *(the* $\log$ *of the slope values) as linear predictor to better fulfill the model assumptions of the liner mixed models. No model is statistically significant at* $\alpha = 5\%$.

	Richness		Evenness	
buffer radius [km]	log(slope)	p-value	log(slope)	p-value
0.1	0.170	0.185	0.314	0.122
0.5	0.040	0.835	0.364	0.163
1	-0.055	0.806	0.302	0.303
2	-0.068	0.786	0.201	0.526
5	-0.183	0.501	0.006	0.989
10	-0.349	0.214	-0.153	0.651

instance, Elmendorf et al. (2012a) homogenize the data. They state: "For a simple index of change that is comparable across sites, we summarized the direction of change for each growth form at each site based on the sign of the site-specific linear trend over time." Related is the heterogeneity in the spatial sampling designs, which leads to huge differences in the spatial sizes of sites (Table 1).

Landsat data

- The Landsat data are sparse in time at many ITEX site locations (Figure A2). Remarkably, there are many years for which no observation were available for certain sites. Also, the number of available images varied between sites. All this factors influenced the construction of the MVCs and may have lead to datasets that are incomparable between sites (Figure A4 and A5).
- The quality information based on the CFmask algorithm helped to remove pixels with low quality. However, small objects (e. g. , lakes) may not be recognized by the CFmask algorithm and influence the observed values. Moreover, it is known that the NDVI does not only measure vegetation, but is also sensitive to variation of the type of vegetation, litter, bare ground, and soil-moisture making its interpretation difficult (Tucker et al., 2005).
- The spatial resolution of the Landsat images is $\approx 30\,\mathrm{m}$, which could be too large to detect features relevant for biodiversity measured at meter scale.

ASTER GDEM

- The spatial resolution of the ASTER GDEM is of the order of one arc second, which could be too large to detect features relevant for biodiversity measured at meter scale.

Acknowledgments

We thank Miguel Jaleshon for support with regard to the preparation of the ITEX abundance data and Jeff Kerby, Anders Michelsen, Greg Henry, Isla Myers-Smith and Ester Lévesque for sending missing spatial information in the ITEX dataset. We thank Jacqueline Oehri for interesting discussions about biodiversity theories. We acknowledge support of the University of Zurich Research Priority Program (URPP) on "Global Change and Biodiversity."

5 Bibliography

ACIA. Arctic Climate Impact Assessment: Scientific Report. Cambridge University Press, 2005. URL http://www.acia.uaf.edu.

Barnhart, K. R., Miller, C. R., Overeem, I., and Kay, J. E. Mapping the future expansion of Arctic open water. Nature Climate Change, 6(3):280–285, 2016. ISSN 1758-678X. doi:10.1038/nclimate2848. Letter.

Bates, D., Mächler, M., Bolker, B., and Walker, S. Fitting linear mixed-effects models using lme4. Journal of Statistical Software, 67(1):1–48, 2015. doi:10.18637/jss.v067.i01.

Blok, D., Heijmans, M. M. P. D., Schaepman-Strub, G., Kononov, A. V., Maximov, T. C., and Berendse, F. Shrub expansion may reduce summer permafrost thaw in Siberian tundra. Global Change Biology, 16(4):1296–1305, 2010. ISSN 1365-2486. doi:10.1111/j.1365-2486.2009.02110.x.

Blok, D., Schaepman-Strub, G., Bartholomeus, H., Heijmans, M. M. P. D., Maximov, T. C., and Berendse, F. The response of Arctic vegetation to the summer climate: relation between shrub cover, NDVI, surface albedo and temperature. Environmental Research Letters, 6(3):035502, 2011. doi:10.1088/1748-9326/6/3/035502.

Chapin, F. S., Sturm, M., Serreze, M. C., McFadden, J. P., Key, J. R., Lloyd, A. H., McGuire, A. D., Rupp, T. S., Lynch, A. H., Schimel, J. P., Beringer, J., Chapman, W. L., Epstein, H. E., Euskirchen, E. S., Hinzman, L. D., Jia, G., Ping, C.-L., Tape, K. D., Thompson, C. D. C., Walker, D. A., and Welker, J. M. Role of land-surface changes in Arctic summer warming. Science, 310(5748):657–660, 2005. ISSN 0036-8075. doi:10.1126/science.1117368.

Colwell, R. K. Biodiversity: Concepts, Patterns, and Measurement, pages 257–263. Princeton University Press, 2009. doi:10.1515/9781400833023.257.

Department of the Interior U.S. Geological Survey (USGS). LANDSAT 4–7 surface reflectance (LEDAPS) product. Product Guide, 2017. URL https://landsat.usgs.gov/sites/default/files/documents/ledaps_product_guide.pdf.

Elmendorf, S. C., Henry, G. H. R., Hollister, R. D., Bjork, R. G., Boulanger-Lapointe, N., Cooper, E. J., Cornelissen, J. H. C., Day, T. A., Dorrepaal, E., Elumeeva, T. G., Gill, M., Gould, W. A., Harte, J., and et al. Plot-scale evidence of tundra vegetation change and links to recent summer warming. Nature Climate Change, 2(6):453–457, 2012a. ISSN 1758-678X. doi:10.1038/nclimate1465.

Elmendorf, S. C., Henry, G. H. R., Hollister, R. D., Bjrk, R. G., Bjorkman, A. D., Callaghan, T. V., Collier, L. S., Cooper, E. J., Cornelissen, J. H. C., Day, T. A., Fosaa, A. M., Gould, W. A., Grtarsdttir, J., Harte, J., Hermanutz, L., Hik, D. S., and others. Global assessment of experimental climate warming on tundra vegetation: heterogeneity over space and time. Ecology Letters, 15(2):164–175, 2012b. ISSN 1461-0248. doi:10.1111/j.1461-0248.2011.01716.x.

Foga, S., Scaramuzza, P. L., Guo, S., Zhu, Z., Jr, R. D. D., Beckmann, T., Schmidt, G. L., Dwyer, J. L., Hughes, M. J., and Laue, B. Cloud detection algorithm comparison and validation for operational Landsat data products. Remote Sensing of Environment, 194:379–390, 2017. ISSN 0034-4257. doi:10.1016/j.rse.2017.03.026.

Gillman, L. N. and Wright, S. D. The influence of productivity on the species richness of plants: a critical assessment. Ecology, 87(5):1234–1243, 2006. ISSN 1939-9170. doi:10.1890/0012-9658(2006)87[1234:TIOPOT]2.0.CO;2.

Google Earth Engine Team. Google earth engine: A planetary-scale geo-spatial analysis platform, 2016. URL https://earthengine.google.com.

Hill, M. O. Diversity and evenness: A unifying notation and its consequences. Ecology, 54(2):427–432, 1973. ISSN 1939-9170. doi:10.2307/1934352.

Hinzman, L. D., Deal, C. J., McGuire, A. D., Mernild, S. H., Polyakov, I. V., and Walsh, J. E. Trajectory of the Arctic as an integrated system. Ecological Applications, 23(8):1837–1868, 2013. ISSN 1939-5582. doi:10.1890/11-1498.1.

Hothorn, T., Möst, L., and Bühlmann, P. Most Likely Transformations. ArXiv e-prints, 2015. URL http://adsabs.harvard.edu/abs/2015arXiv150806749H.

Hothorn, T. Most Likely Transformations: The mlt package, 2017. URL https://cran.r-project.org/package=mlt.docreg. R package vignette version 0.1-5.

Hothorn, T., Kneib, T., and Bühlmann, P. Conditional transformation models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(1):3–27, 2014. ISSN 1467-9868. doi:10.1111/rssb.12017.

Huston, M. A. Disturbance, productivity, and species diversity: empiricism vs. logic in ecological theory. Ecology, 95(9): 2382–2396, 2014. ISSN 1939-9170. doi:10.1890/13-1397.1.

IPCC. Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2013. ISBN 978-1-107-66182-0. doi:10.1017/CBO9781107415324. URL www.climatechange2013.org.

Jones, H. G. and Vaughan, R. A. Remote sensing of vegetation: principles, techniques, and applications. Oxford University Press, 2010. ISBN 978-0-19-920779-4.

Jones, T. G., Coops, N. C., Gergel, S. E., and Sharma, T. Employing measures of heterogeneity and an object-based approach to extrapolate tree species distribution data. Diversity, 6(3):396–414, 2014. ISSN 1424-2818. doi:10.3390/d6030396.

Limpens, J., Berendse, F., Blodau, C., Canadell, J. G., Freeman, C., Holden, J., Roulet, N., Rydin, H., and Schaepman-Strub, G. Peatlands and the carbon cycle: from local processes to global implications—a synthesis. Biogeosciences, 5(5): 1475–1491, 2008. doi:10.5194/bg-5-1475-2008.

Masek, J. G., Vermote, E. F., Saleous, N. E., Wolfe, R., Hall, F. G., Huemmrich, K. F., Gao, F., Kutler, J., and Lim, T.-K. A Landsat surface reflectance dataset for North America, 1990–2000. IEEE Geoscience and Remote Sensing Letters, 3 (1):68–72, 2006. doi:10.1109/LGRS.2005.857030.

Mittelbach, G. G., Steiner, C. F., Scheiner, S. M., Gross, K. L., Reynolds, H. L., Waide, R. B., Willig, M. R., Dodson, S. I., and Gough, L. What is the observed relationship between species richness and productivity? Ecology, 82(9):2381–2396, 2001. ISSN 1939-9170. doi:10.1890/0012-9658(2001)082[2381:WITORB]2.0.CO;2.

Myers-Smith, I. H., Elmendorf, S. C., Beck, P. S. A., Wilmking, M., Hallinger, M., Blok, D., Tape, K. D., Rayback, S. A., Macias-Fauria, M., Forbes, B. C., Speed, J. D. M., Boulanger-Lapointe, N., Rixen, C., Levesque, E., Schmidt, N. M., Baittinger, C., Trant, A. J., Hermanutz, L., Collier, L. S., Dawes, M. A., Lantz, T. C., Weijers, S., Jorgensen, R. H., Buchwal, A., Buras, A., Naito, A. T., Ravolainen, V., Schaepman-Strub, G., Wheeler, J. A., Wipf, S., Guay, K. C., Hik, D. S., and Vellend, M. Climate sensitivity of shrub growth across the tundra biome. Nature Climate Change, 5(9): 887–891, 2015. ISSN 1758-678X. doi:10.1038/nclimate2697. Letter.

NASA LP DAAC, METI. Routine ASTER global digital elevation model ASTGTM, 2011.

Oberbauer, S. F., Elmendorf, S. C., Troxler, T. G., Hollister, R. D., Rocha, A. V., Bret-Harte, M. S., Dawes, M. A., Fosaa, A. M., Henry, G. H. R., Hye, T. T., Jarrad, F. C., Jnsdttir, I. S., Klanderud, K., Klein, J. A., Molau, U., Rixen, C., Schmidt, N. M., Shaver, G. R., Slider, R. T., Totland, ., Wahren, C.-H., and Welke, J. M. Phenological response of tundra plants to background climate variation tested using the International Tundra Experiment. Philosophical Transactions of the Royal Society of London B: Biological Sciences, 368(1624), 2013. doi:10.1098/rstb.2012.0481.

Ostendorf, B. and Reynolds, J. F. A model of arctic tundra vegetation derived from topographic gradients. Landscape Ecology, 13(3):187–201, 1998. ISSN 1572-9761. doi:10.1023/A:1007986410048.

R. A Language and Environment for Statistical Computing. R Core Team, R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

Raynolds, M. K., Comiso, J. C., Walker, D. A., and Verbyla, D. Relationship between satellite-derived land surface temperatures, Arctic vegetation types, and NDVI. Remote Sensing of Environment, 112(4):1884–1894, 2008. ISSN 0034-4257. doi:10.1016/j.rse.2007.09.008.

Schaepman-Strub, G., Iturrate, M., and Furrer, R. Towards assessing biodiversity feedbacks to climate in the Arctic - future application of the AVA. In Arctic Vegetation Archive (AVA) Workshop, 2013. CAFF, Krakow, Poland, proceeding Series Report Nr. 10.

Schelldorfer, J., Meier, L., and Bühlmann, P. GLMMLasso: An algorithm for high-dimensional generalized linear mixed models using $l1$-penalization. Journal of Computational and Graphical Statistics, 23(2):460–477, 2014. doi:10.1080/10618600.2013.773239.

Serreze, M. C., Walsh, J. E., Chapin, F. S., Osterkamp, T., Dyurgerov, M., Romanovsky, V., Oechel, W. C., Morison, J., Zhang, T., and Barry, R. G. Observational evidence of recent change in the northern high-latitude environment. Climatic Change, 46(1):159–207, 2000. ISSN 1573-1480. doi:10.1023/A:1005504031923.

Stein, A., Gerstner, K., and Kreft, H. Environmental heterogeneity as a universal driver of species richness across taxa, biomes and spatial scales. Ecology Letters, 17(7):866–880, 2014. ISSN 1461-0248. doi:10.1111/ele.12277.

Tuanmu, M.-N. and Jetz, W. A global, remote sensing-based characterization of terrestrial habitat heterogeneity for biodiversity and ecosystem modelling. Global Ecology and Biogeography, 24(11):1329–1339, 2015. ISSN 1466-8238. doi:10.1111/geb.12365.

Tucker, C. J., Pinzon, J. E., Brown, M. E., Slayback, D. A., Pak, E. W., Mahoney, R., Vermote, E. F., and Saleous, N. E. An extended AVHRR 8 km NDVI dataset compatible with MODIS and SPOT vegetation NDVI data. International Journal of Remote Sensing, 26(20):4485–4498, 2005. doi:10.1080/01431160500168686.

Virtanen, R., Grytnes, J.-A., Lenoir, J., Luoto, M., Oksanen, J., Oksanen, L., and Svenning, J.-C. Productivity–diversity patterns in Arctic tundra vegetation. Ecography, 36(3):331–341, 2013. ISSN 1600-0587. doi:10.1111/j.1600-0587.2012.07903.x.

Walker, D. A., Epstein, H. E., Jia, G. J., Balser, A., Copass, C., Edwards, E. J., Gould, W. A., Hollingsworth, J., Knudson, J., Maier, H. A., Moody, A., and Raynolds, M. K. Phytomass, LAI, and NDVI in northern Alaska: Relationships to summer warmth, soil ph, plant functional types, and extrapolation to the circumpolar Arctic. Journal of Geophysical Research: Atmospheres, 108(D2), 2003. ISSN 2156-2202. doi:10.1029/2001JD000986.

Zimov, S. A., Schuur, E. A. G., and Chapin, F. S. Permafrost and the global carbon budget. Science, 312(5780):1612–1613, 2006. ISSN 0036-8075. doi:10.1126/science.1128908.

Appendix

The following material gives more information on the performed analyses. Additional figures and tables are given in Appendix A and B, respectively. Appendix C present a more mathematical description of mixed effects models. Eventually, Appendix D presents an approach to relate field based measurements with satellite data based on Most Likely Transformations (MLT) models.

A Figures

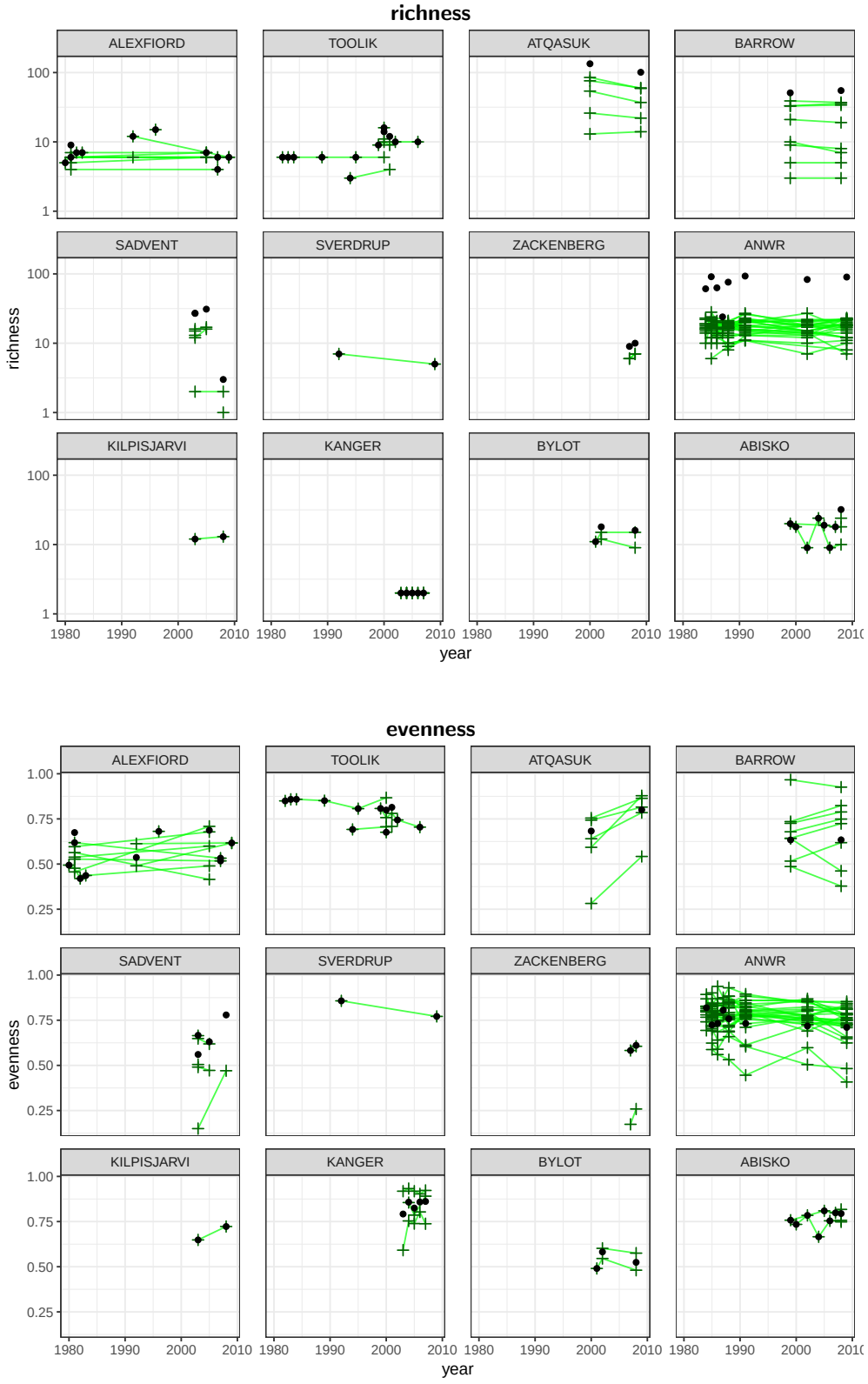


Figure A1: Temporal variation of the richness (top) and evenness (bottom) biodiversity indices. The names of the sites are indicated at the top of each panel. Black dots represent the indices calculated at site level, whereas the connected green crosses represent the indices calculated at sub-site level.

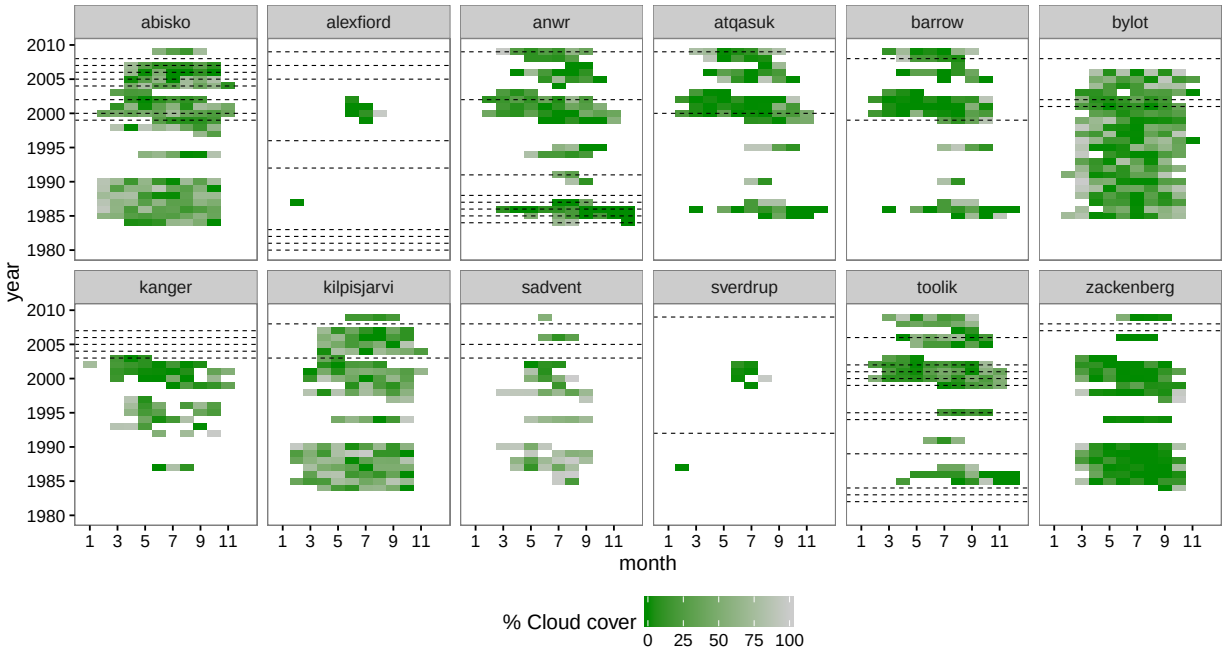


Figure A2: Available Landsat image covering the indicated sites. Horizontal lines indicate the availability of ITEX plot data.

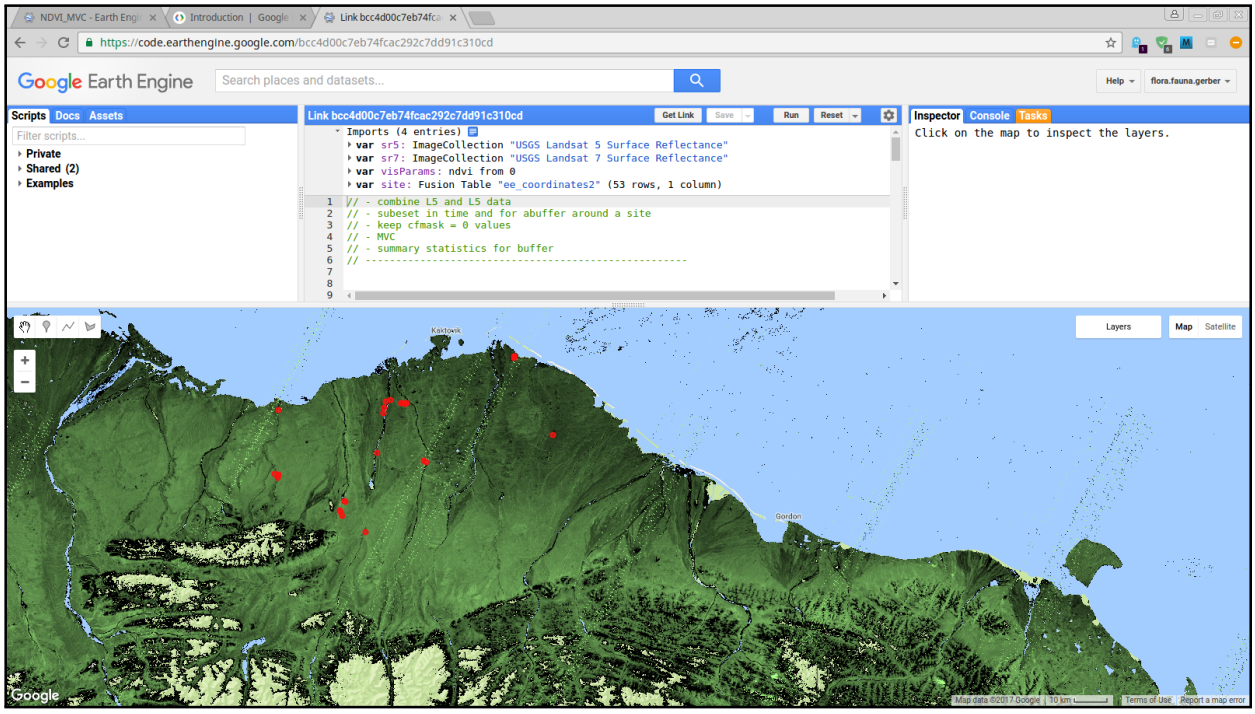


Figure A3: Screenshot of the Google Earth Engine environment used to create the Landsat MVCs. The red dots indicate locations of ITEX sub-sites. Date of the screenshot: Mai 5, 2017.

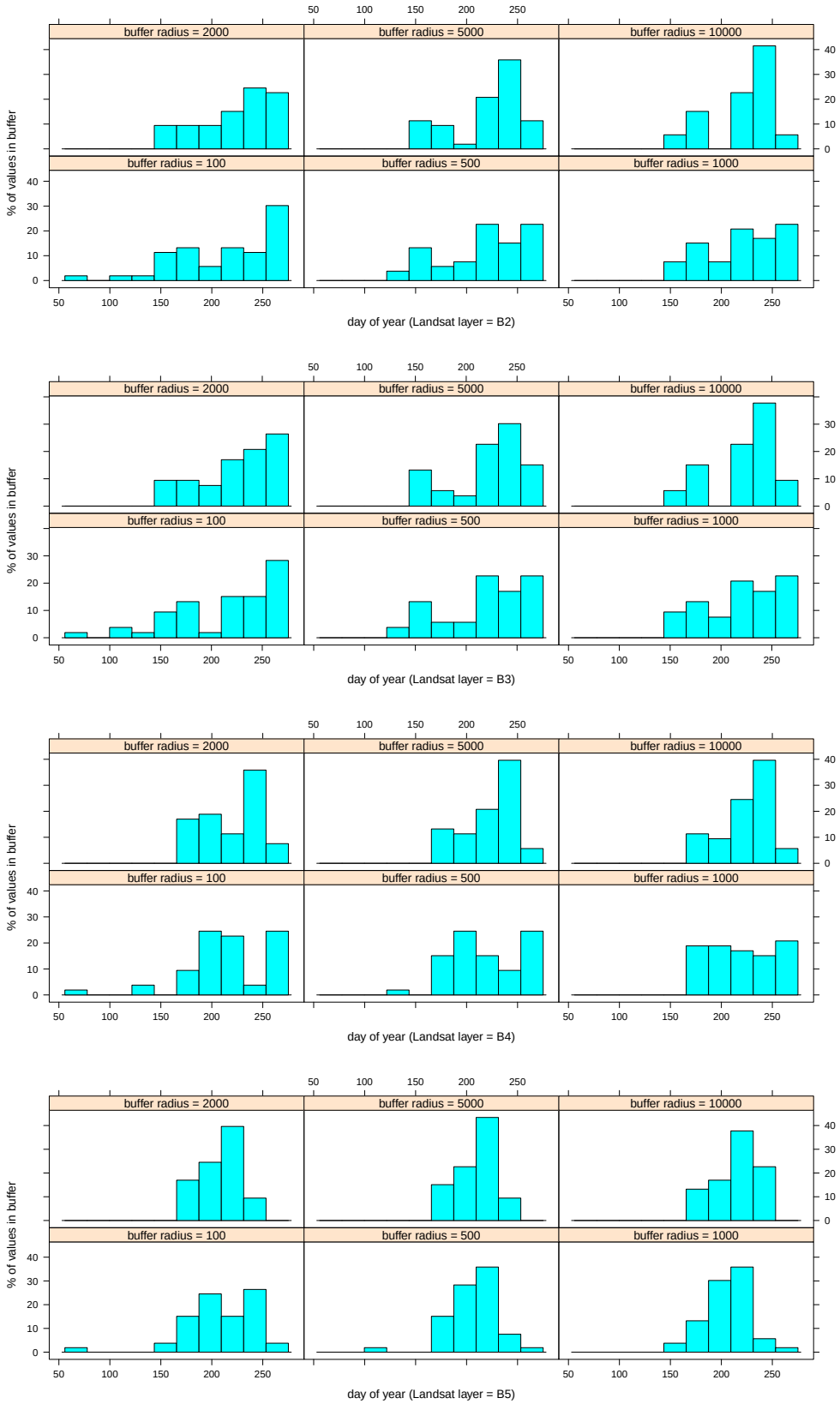


Figure A4: Histograms of the days of the year (x-axis) of the considered MVCs from Landsat. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

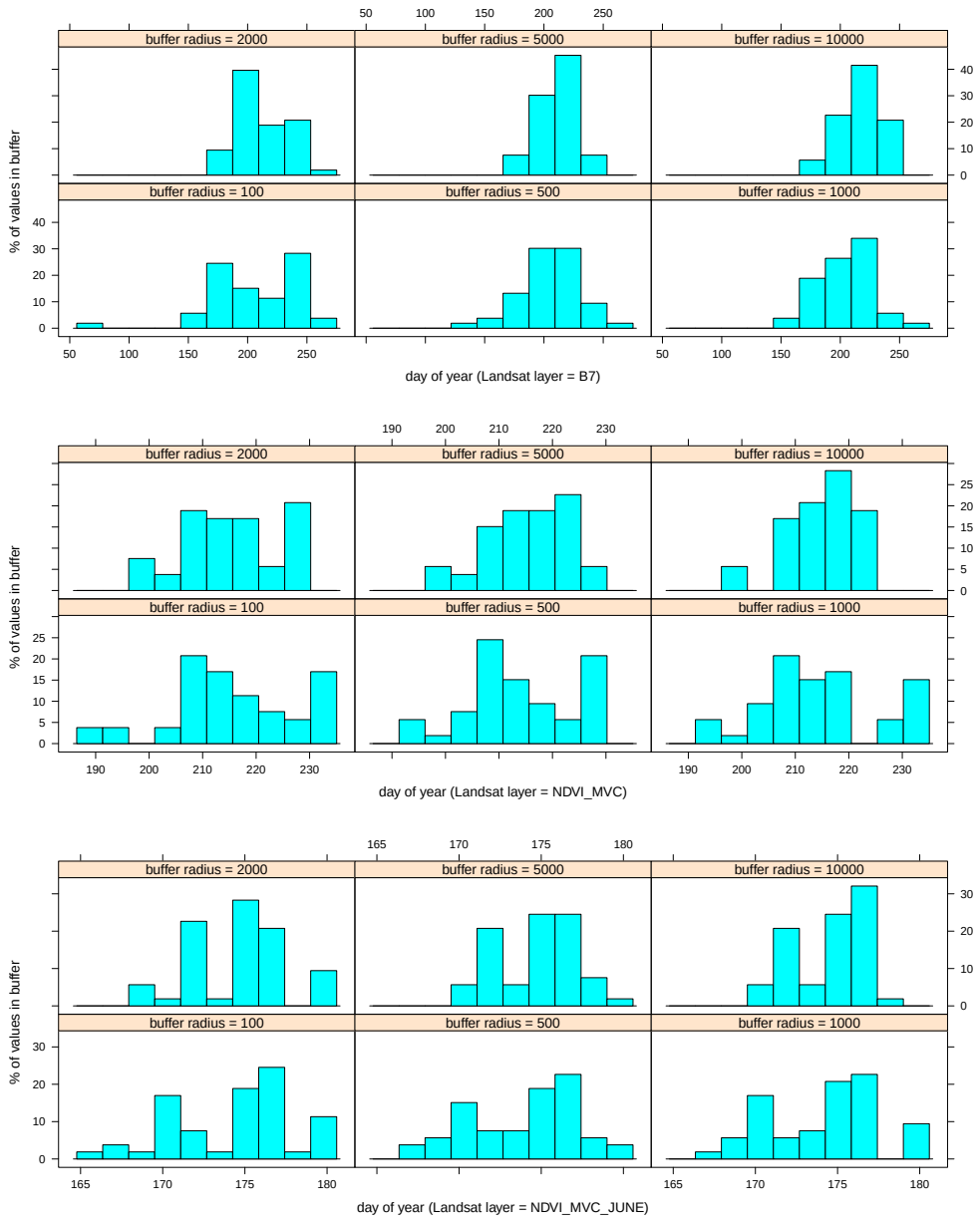


Figure A5: Histograms of the days of the year (x-axis) of the considered MVCs from Landsat. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

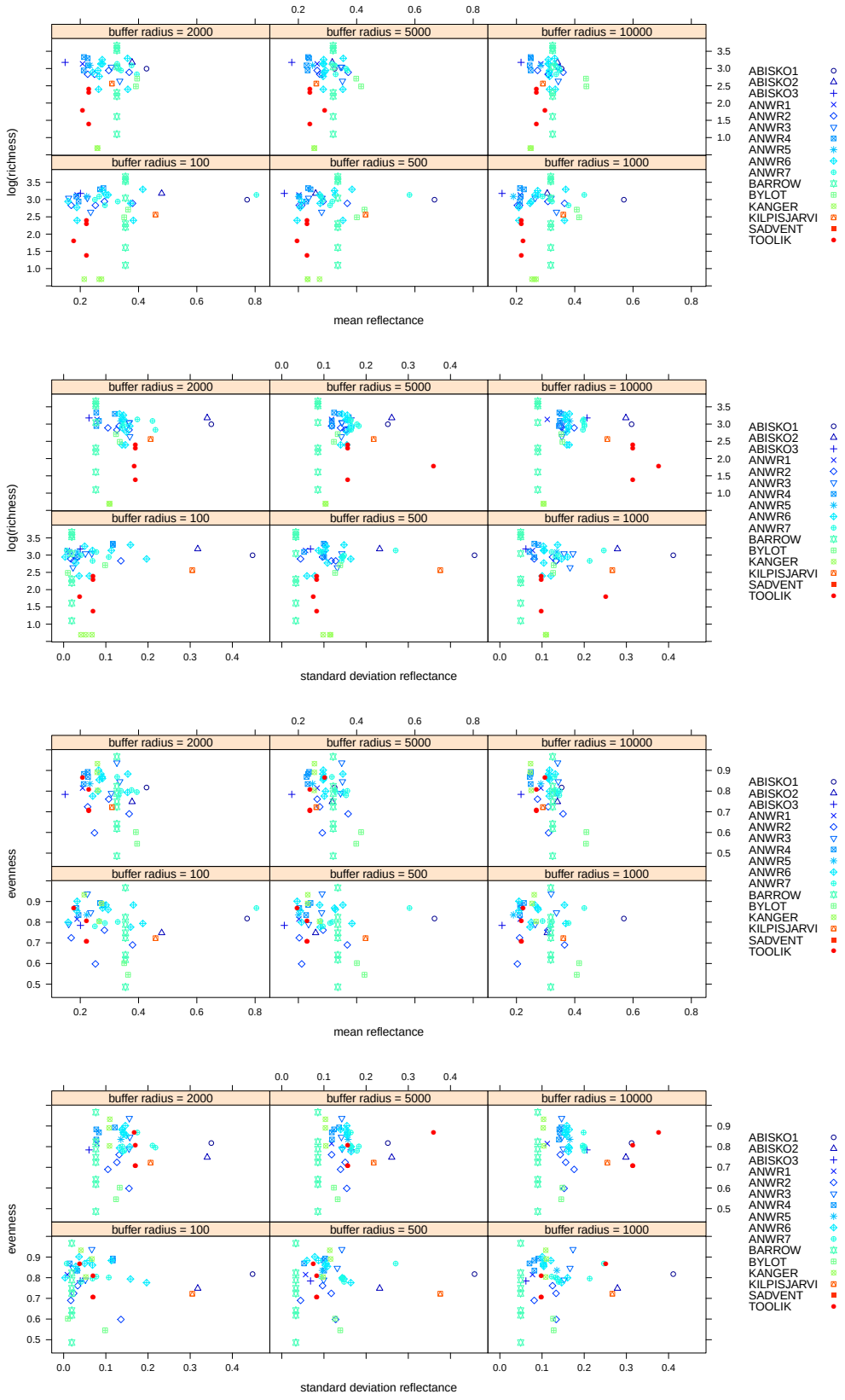


Figure A6: Scatter plots with summary statistics of the Landsat MVC from reflectance factors of band B2 (x-axis) and the corresponding log(richness) and evenness indices (y-axis). The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

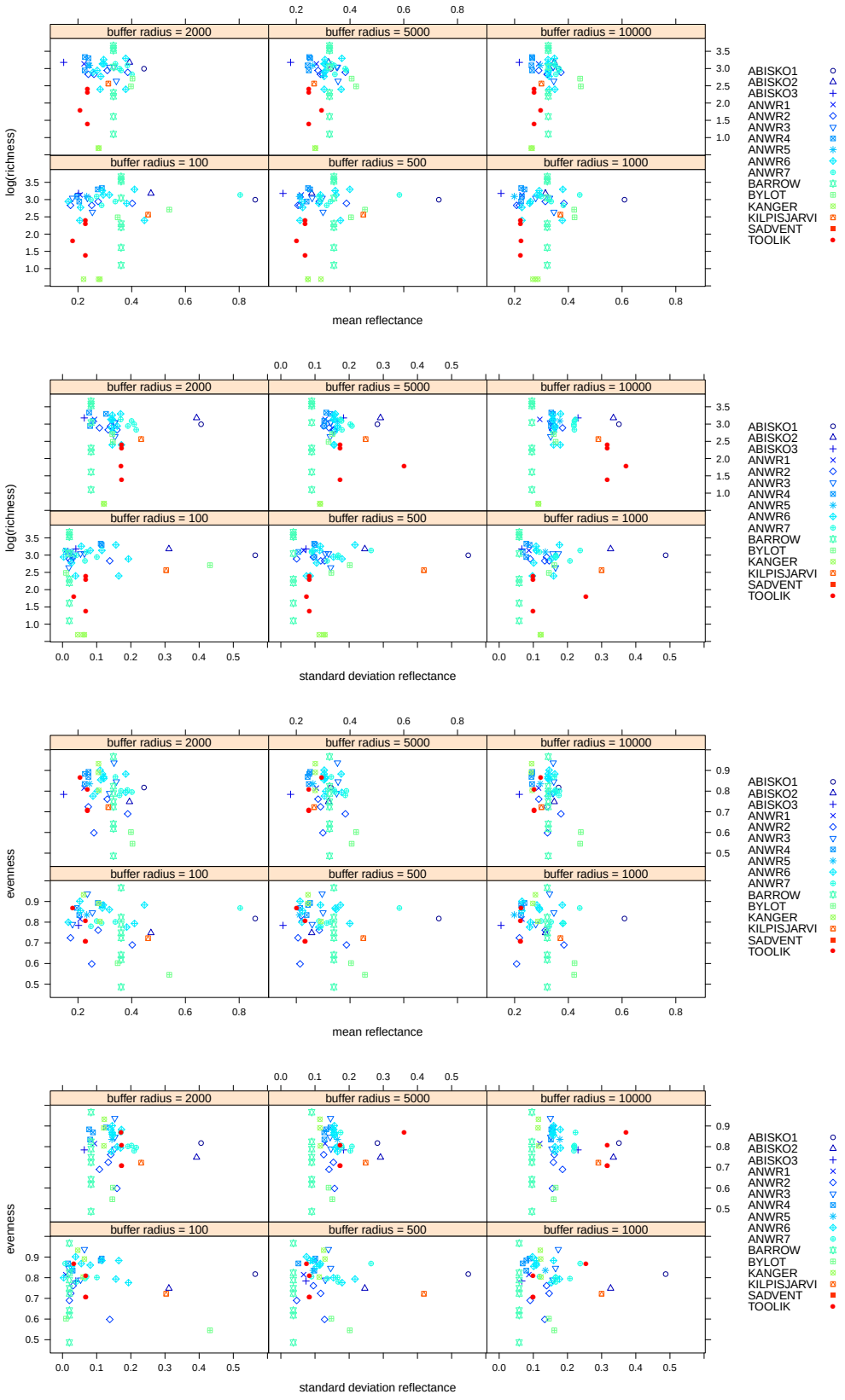


Figure A7: Scatter plots with summary statistics of the Landsat MVC from reflectance factors of band B3 (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

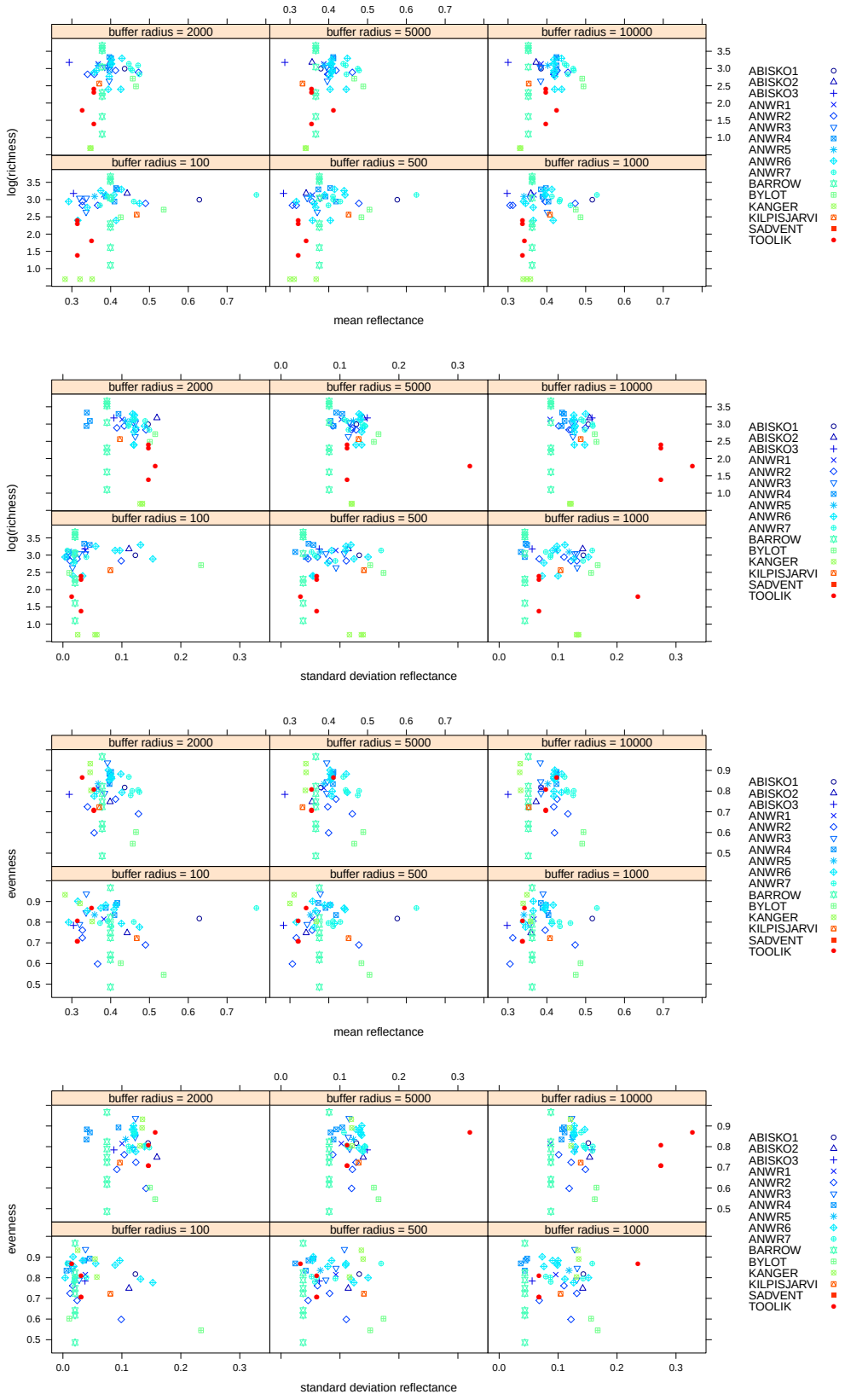


Figure A8: Scatter plots with summary statistics of the Landsat MVC from reflectance factors of band B4 (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

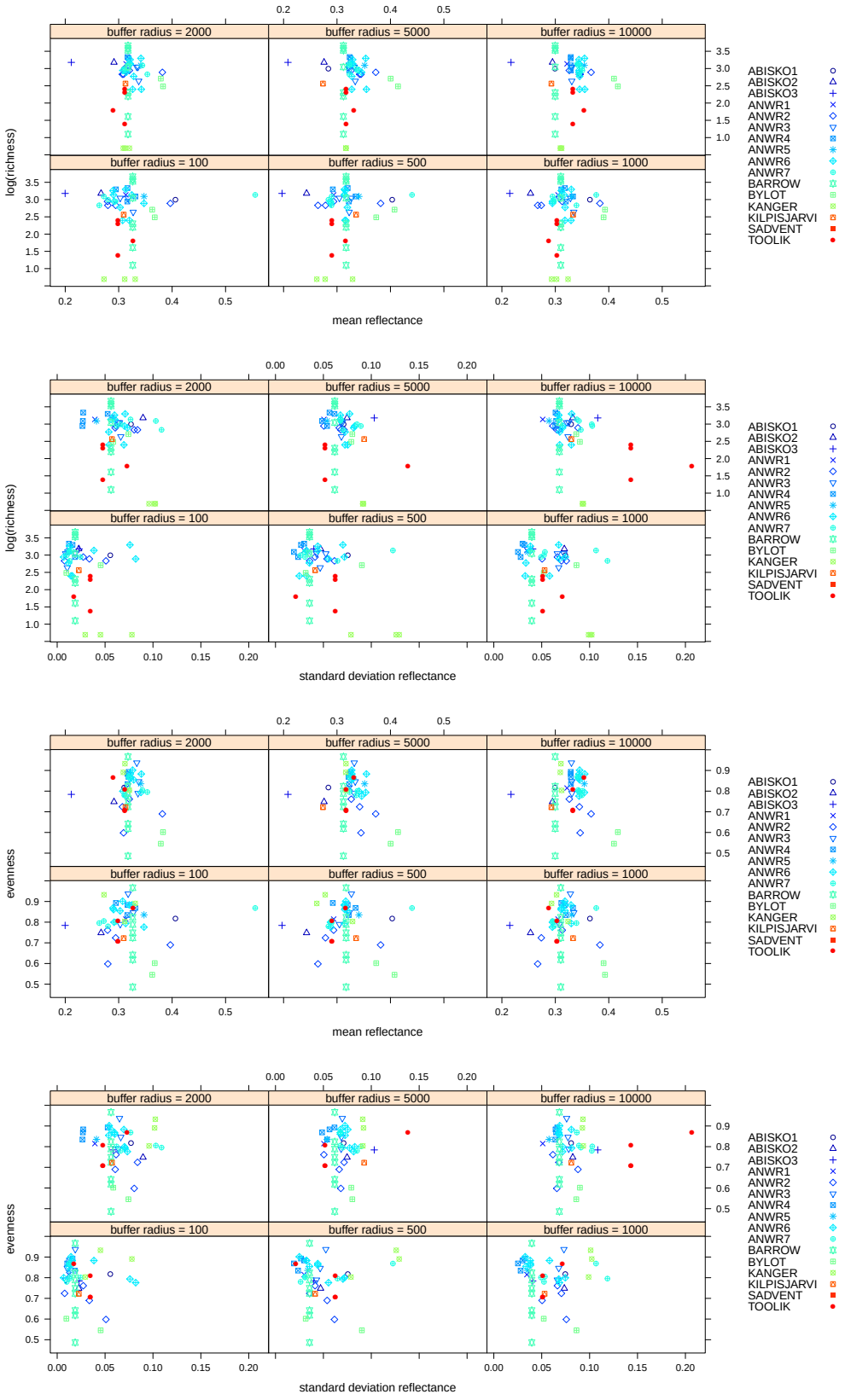


Figure A9: Scatter plots with summary statistics of the Landsat MVC from reflectance factors of band B5 (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

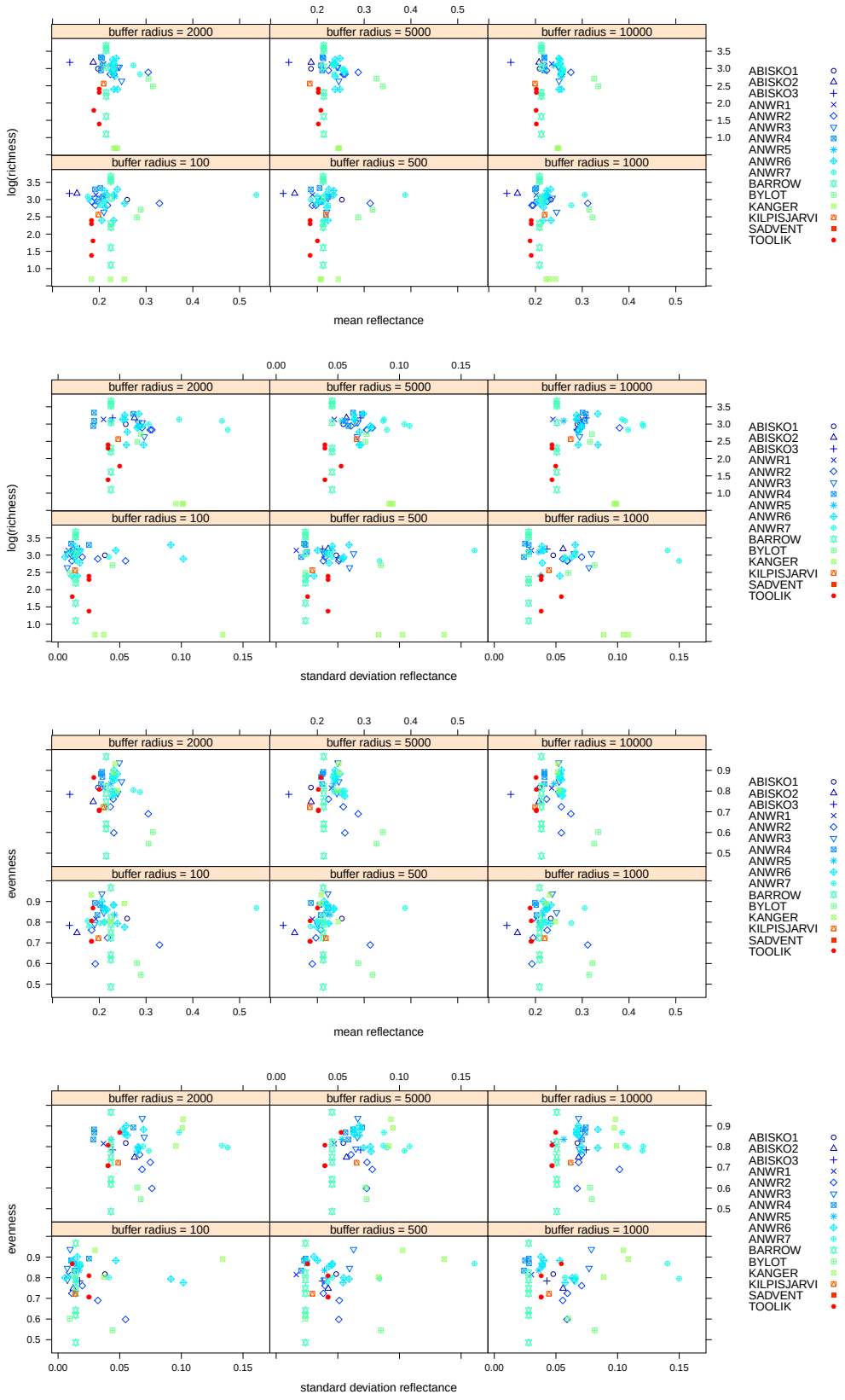


Figure A10: Scatter plots with summary statistics of the Landsat MVC from reflectance factors of band B7 (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

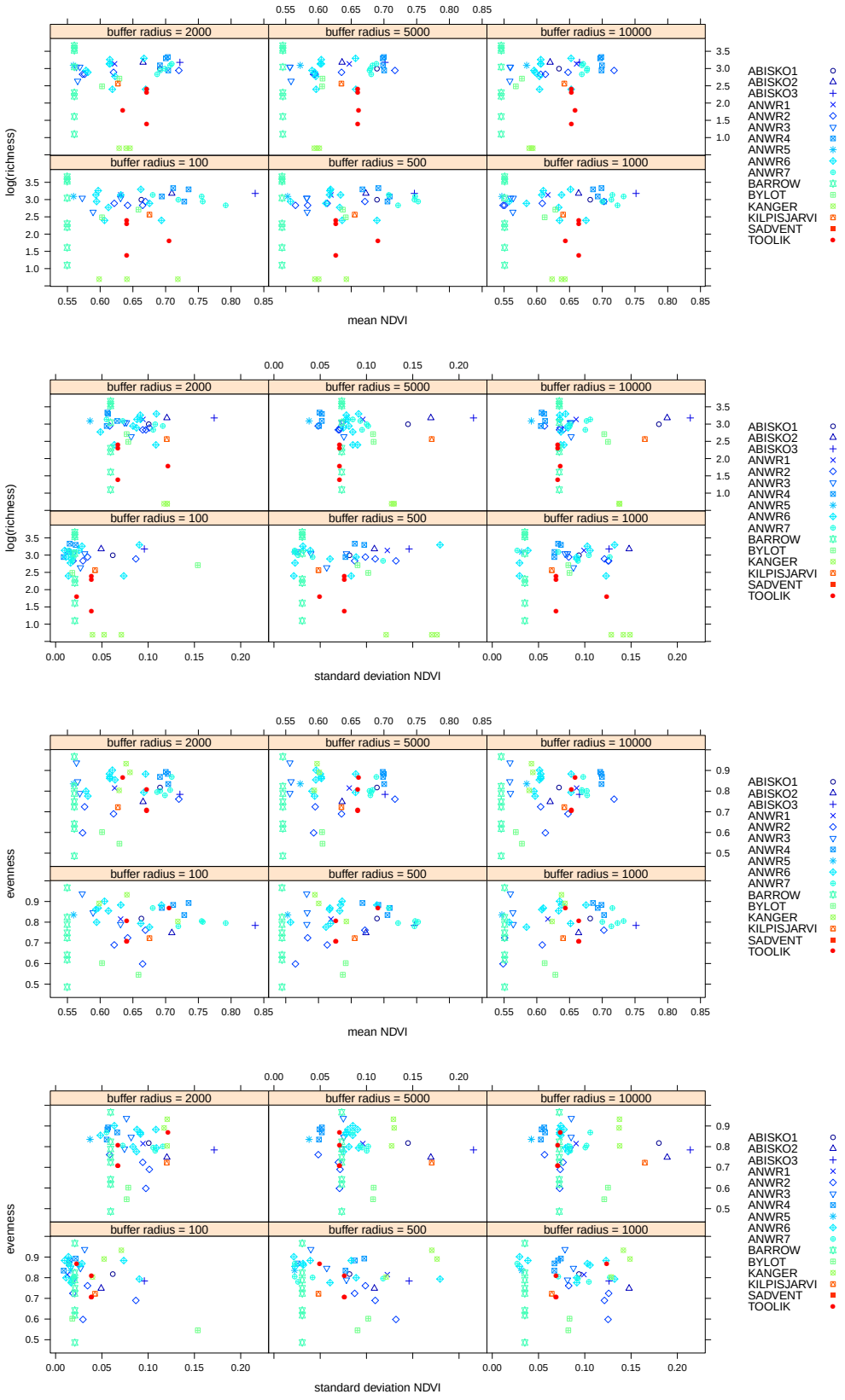


Figure A11: Scatter plots with summary statistics of the Landsat MVC of NDVI values (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

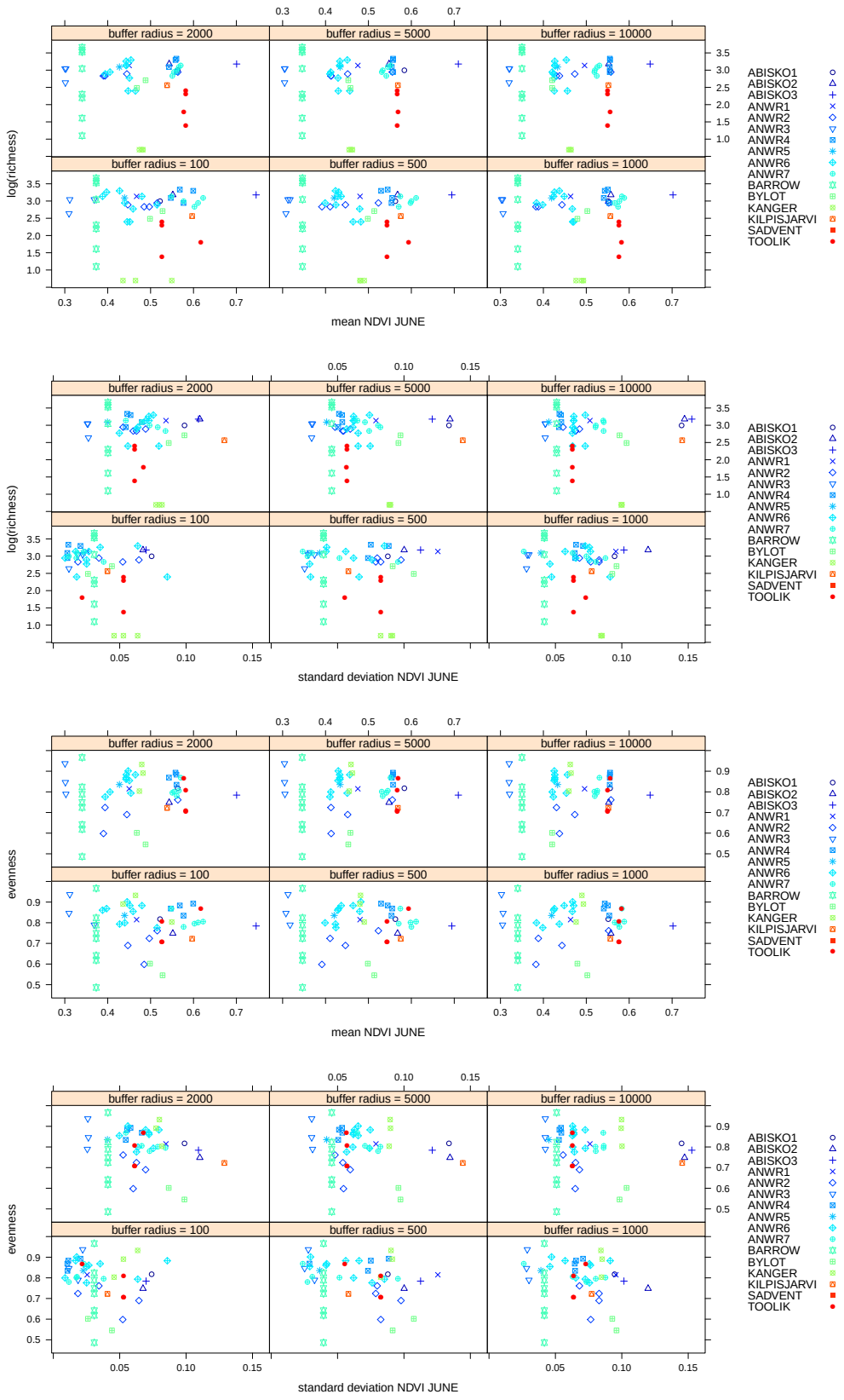


Figure A12: Scatter plots with summary statistics of the Landsat MVC created with NDVI values from June (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the MVC images are indicated at the top of each panel.

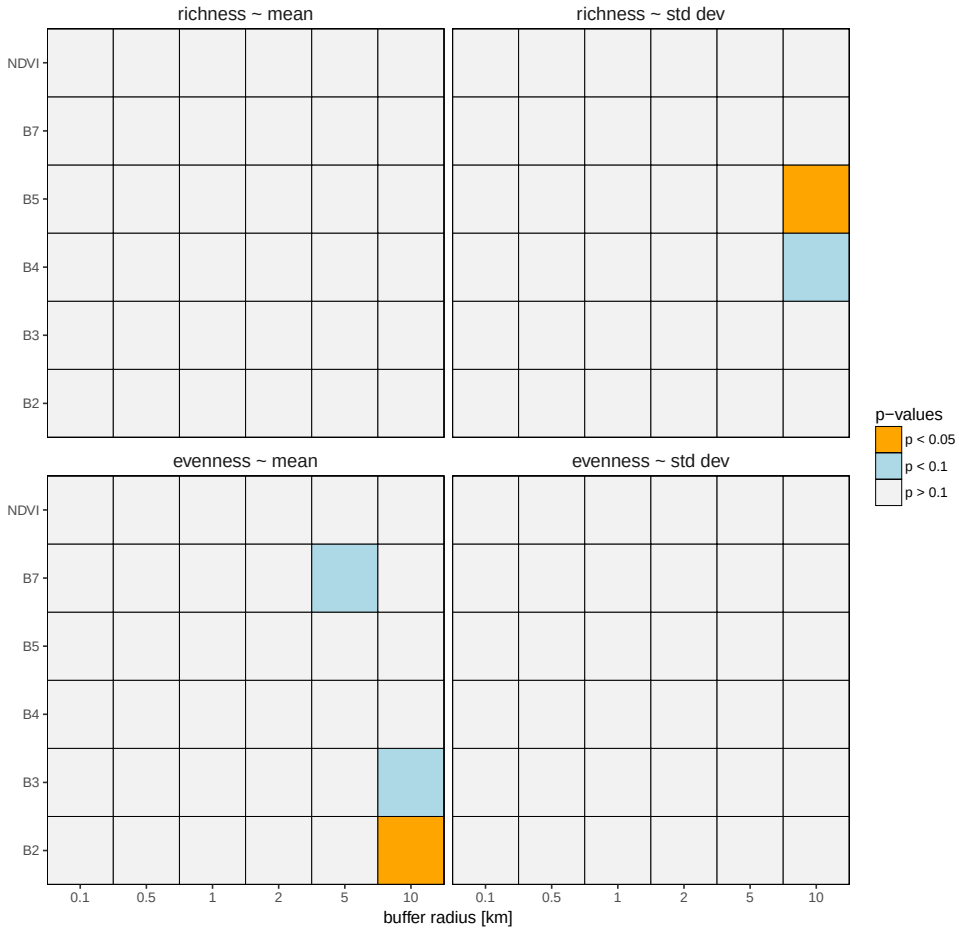


Figure A13: Pattern of significant predictors of the GLMMs for Landsat layers. Top panels: predictors `mean()` and `stdDev()` for the models with log(richness) as response variable listed in Table A1. Bottom panels: predictors `mean()` and `stdDev()` for the models with evenness as response variable listed in Table A2.

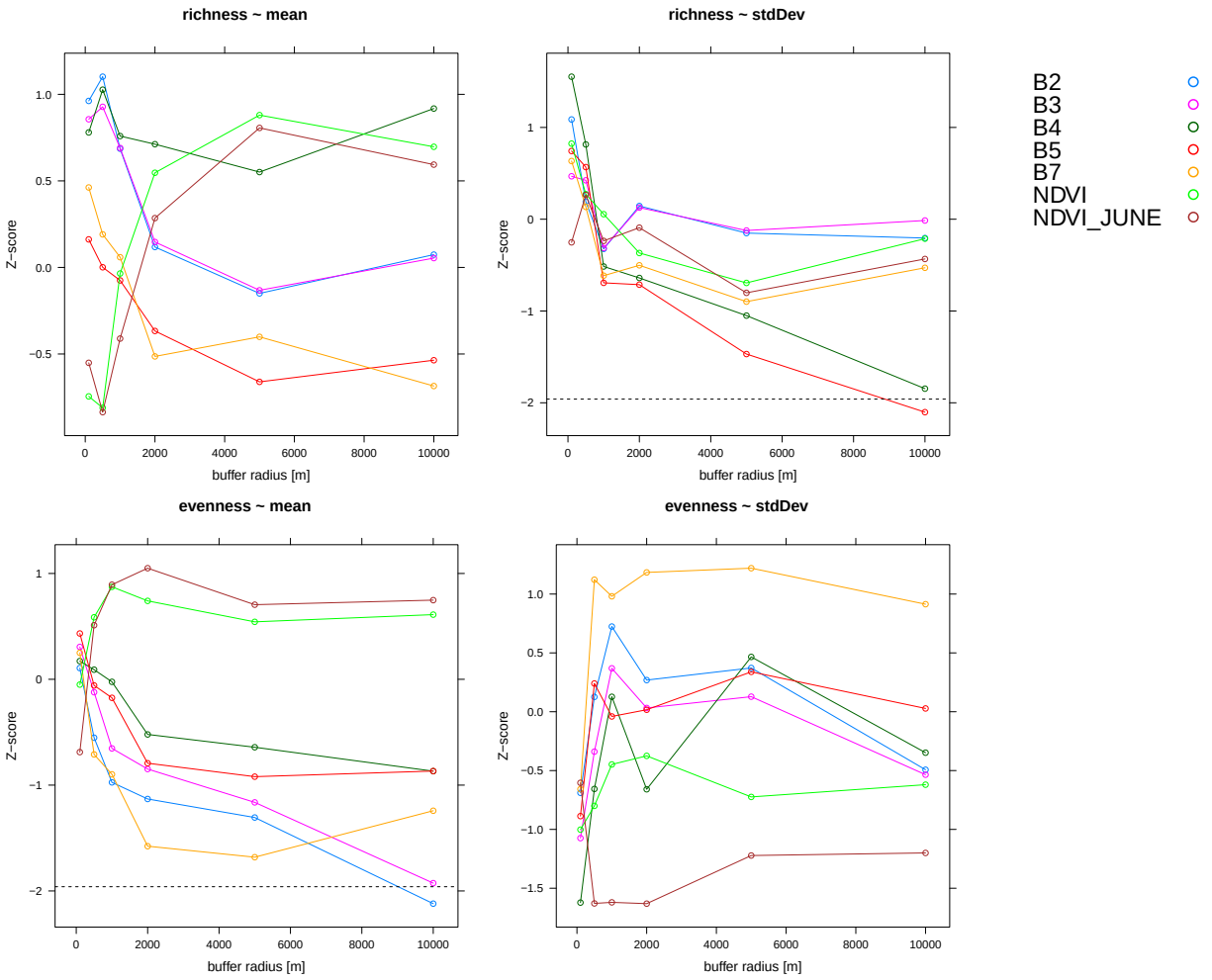


Figure A14: Z-score values of the Landsat based predictors explaining sub-site biodiversity. Each dot represents a model predictor, which is significant at $\alpha = 5\%$ if the dot is below -1.96 or above 1.96. Top row: Z-score values of the generalized linear mixed model having the sub-site species richness as response variable and the mean (left) and the standard deviation (right) from reflectance factors of the indicated Landsat bands extracted at the indicated buffer size (scale) as predictors. Bottom row: Z-score values of the linear mixed model having the sub-site species evenness as response variable and the mean (left) and the standard deviation (right) of reflectance factors from the indicated Landsat bands extracted at the indicated buffer size (scale) as predictors.

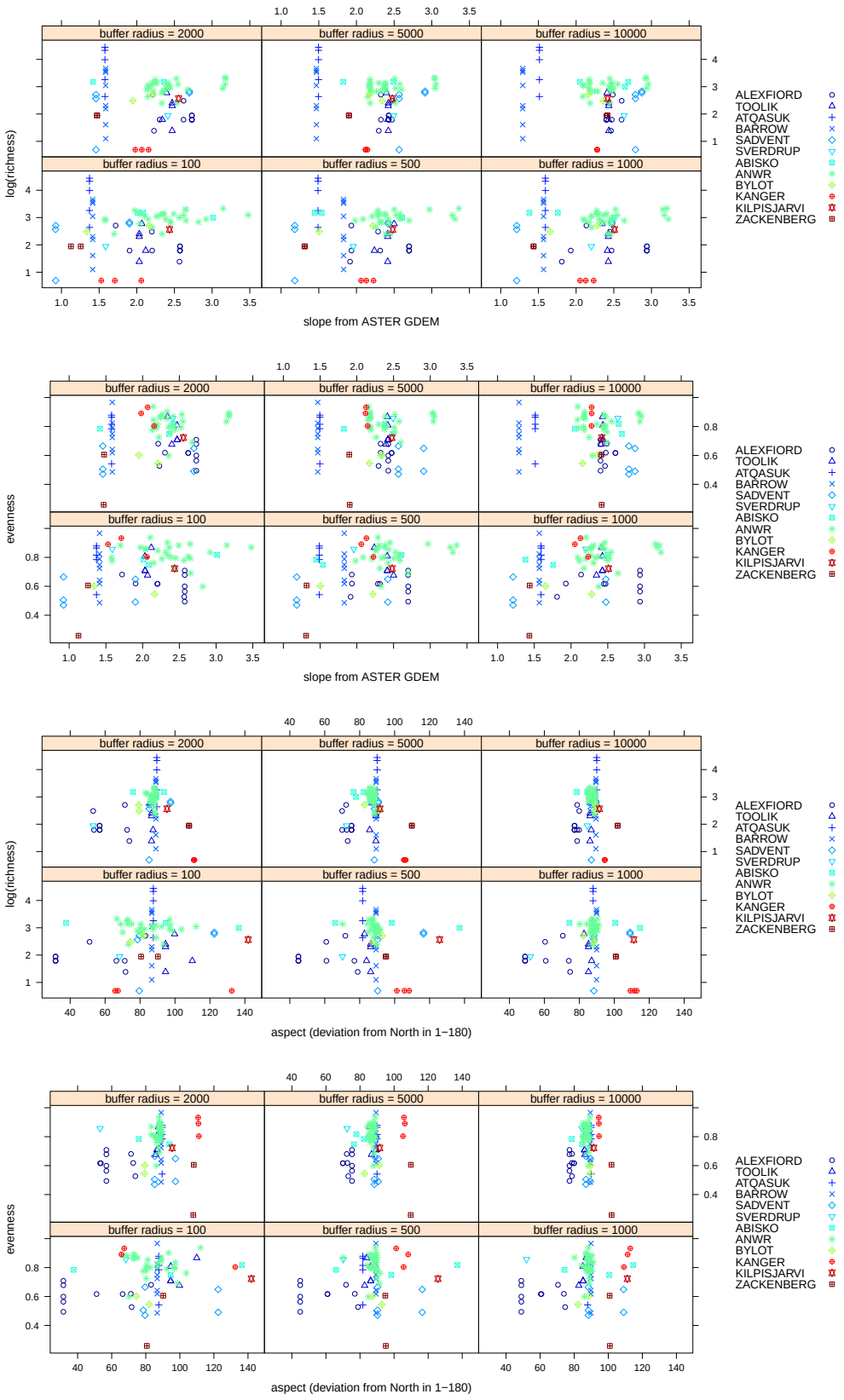


Figure A15: Scatter plots with summary statistics of the ASTER GDEM based slope and aspect values (x-axis) and the corresponding log(richness) and evenness indices on the y-axis. The buffer radii in meters used to summarize the images are indicated at the top of each panel.

B Tables

Table A1: List of the GLMMs fitted to test whether the richness index can be explained with summary statistics of reflectance factors of a single Landsat bands. The single significant predictor at $\alpha = 5\%$ is underlined and marked red (model 24).

no.	model formula	buffer radius [km]	distribution
1	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	0.1	Poisson
2	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	0.5	Poisson
3	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	1	Poisson
4	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	2	Poisson
5	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	5	Poisson
6	`log(RICHNESS) ~ mean(B2) + stdDev(B2) + (1|SITE)`	10	Poisson
7	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	0.1	Poisson
8	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	0.5	Poisson
9	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	1	Poisson
10	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	2	Poisson
11	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	5	Poisson
12	`log(RICHNESS) ~ mean(B3) + stdDev(B3) + (1|SITE)`	10	Poisson
13	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	0.1	Poisson
14	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	0.5	Poisson
15	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	1	Poisson
16	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	2	Poisson
17	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	5	Poisson
18	`log(RICHNESS) ~ mean(B4) + stdDev(B4) + (1|SITE)`	10	Poisson
19	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	0.1	Poisson
20	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	0.5	Poisson
21	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	1	Poisson
22	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	2	Poisson
23	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	5	Poisson
24	`log(RICHNESS) ~ mean(B5) + stdDev(B5) + (1|SITE)`	10	Poisson
25	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	0.1	Poisson
26	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	0.5	Poisson
27	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	1	Poisson
28	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	2	Poisson
29	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	5	Poisson
30	`log(RICHNESS) ~ mean(B7) + stdDev(B7) + (1|SITE)`	10	Poisson
31	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	0.1	Poisson
32	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	0.5	Poisson
33	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	1	Poisson
34	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	2	Poisson
35	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	5	Poisson
36	`log(RICHNESS) ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)`	10	Poisson
37	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	0.1	Poisson
38	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	0.5	Poisson
39	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	1	Poisson
40	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	2	Poisson
41	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	5	Poisson
42	`log(RICHNESS) ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)`	10	Poisson

Table A2: List of the GLMMs fitted to test whether the species evenness index can be explained with summary statistics of reflectance factors of a single Landsat bands. The single significant predictor at $\alpha = 5\%$ is underlined and marked red (model 48).

no.	model formula	buffer radius [km]	distribution
43	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	0.1	Gaussian
44	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	0.5	Gaussian
45	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	1	Gaussian
46	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	2	Gaussian
47	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	5	Gaussian
48	EVENNESS ~ mean(B2) + stdDev(B2) + (1|SITE)	10	Gaussian
49	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	0.1	Gaussian
50	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	0.5	Gaussian
51	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	1	Gaussian
52	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	2	Gaussian
53	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	5	Gaussian
54	EVENNESS ~ mean(B3) + stdDev(B3) + (1|SITE)	10	Gaussian
55	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	0.1	Gaussian
56	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	0.5	Gaussian
57	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	1	Gaussian
58	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	2	Gaussian
59	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	5	Gaussian
60	EVENNESS ~ mean(B4) + stdDev(B4) + (1|SITE)	10	Gaussian
61	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	0.1	Gaussian
62	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	0.5	Gaussian
63	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	1	Gaussian
64	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	2	Gaussian
65	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	5	Gaussian
66	EVENNESS ~ mean(B5) + stdDev(B5) + (1|SITE)	10	Gaussian
67	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	0.1	Gaussian
68	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	0.5	Gaussian
69	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	1	Gaussian
70	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	2	Gaussian
71	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	5	Gaussian
72	EVENNESS ~ mean(B7) + stdDev(B7) + (1|SITE)	10	Gaussian
73	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	0.1	Gaussian
74	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	0.5	Gaussian
75	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	1	Gaussian
76	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	2	Gaussian
77	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	5	Gaussian
78	EVENNESS ~ mean(NDVI) + stdDev(NDVI) + (1|SITE)	10	Gaussian
79	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	0.1	Gaussian
80	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	0.5	Gaussian
81	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	1	Gaussian
82	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	2	Gaussian
83	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	5	Gaussian
84	EVENNESS ~ mean(NDVI_JUNE) + stdDev(NDVI_JUNE) + (1|SITE)	10	Gaussian

Table A3: List of the GLMMs fitted to test whether the species richness index or the species evenness index can be explained with the slope extracted from ASTER GDEM data. We considered the $\log$ *of the slope values as linear predictor to better fulfill the model assumptions of the liner mixed models. None of the models had a significant predictor at* $\alpha = 5\%$.

no.	model formula	buffer radius [km]	distribution
85	`log(RICHNESS) ~ log(slope) + (1|SITE)`	0.1	Poisson
86	`log(RICHNESS) ~ log(slope) + (1|SITE)`	0.5	Poisson
87	`log(RICHNESS) ~ log(slope) + (1|SITE)`	1	Poisson
88	`log(RICHNESS) ~ log(slope) + (1|SITE)`	2	Poisson
89	`log(RICHNESS) ~ log(slope) + (1|SITE)`	5	Poisson
90	`log(RICHNESS) ~ log(slope) + (1|SITE)`	10	Poisson
91	`EVENNESS ~ log(slope) + (1|SITE)`	0.1	Gaussian
92	`EVENNESS ~ log(slope) + (1|SITE)`	0.5	Gaussian
93	`EVENNESS ~ log(slope) + (1|SITE)`	1	Gaussian
94	`EVENNESS ~ log(slope) + (1|SITE)`	2	Gaussian
95	`EVENNESS ~ log(slope) + (1|SITE)`	5	Gaussian
96	`EVENNESS ~ log(slope) + (1|SITE)`	10	Gaussian

Table A4: List of the GLMMs fitted to test whether the species richness index or the species evenness index can be explained with the aspect extracted from ASTER GDEM data. The dataset was divided into groups with small, medium, and large slopes. The aspect was modeled with the predictors `aspectNS` (deviation from North), `aspectNS` (deviation from West), and the interaction of both predictors. None of the models had a significant predictor at $\alpha = 5\%$.

no.	model formula	buffer radius [km]	distribution	slope
97	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	0.1	Poisson	small
98	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	0.5	Poisson	small
99	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Poisson	small
100	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Poisson	small
101	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Poisson	small
102	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Poisson	small
103	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.1	Poisson	medium
104	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.5	Poisson	medium
105	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Poisson	medium
106	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Poisson	medium
107	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Poisson	medium
108	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Poisson	medium
109	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.1	Poisson	large
110	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.5	Poisson	large
111	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Poisson	large
112	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Poisson	large
113	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Poisson	large
114	`log(RICHNESS) ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Poisson	large
115	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	0.1	Gaussian	small
116	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	0.5	Gaussian	small
117	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Gaussian	small
118	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Gaussian	small
119	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Gaussian	small
120	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Gaussian	small
121	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.1	Gaussian	medium
122	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.5	Gaussian	medium
123	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Gaussian	medium
124	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Gaussian	medium
125	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Gaussian	medium
126	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Gaussian	medium
127	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.1	Gaussian	large
128	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	.5	Gaussian	large
129	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	1	Gaussian	large
130	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	2	Gaussian	large
131	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	5	Gaussian	large
132	`EVENNESS ~ aspectNS + aspectWE + aspectNS:aspectWE + (1|SITE)`	10	Gaussian	large

C Mixed effects models

We introduce the mixed effects models used to analyze the data; see, e. g. , Tables A1–A4. Mixed effects models exhibit fixed and random effects in order to capture correlation structures in the data. We consider the simple random effects model (without fixed effect)

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij} \quad \text{with} \quad \varepsilon_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \tag{2}$$

where $\alpha_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\alpha^2)$, and α_i and ε_{ij} are independent. For that model the expected value is

$$\mathrm{E}(Y_{ij}) = \mathrm{E}(\mu + \alpha_i + \varepsilon_{ij}) = \mathrm{E}(\mu) + \mathrm{E}(\alpha_i) + \mathrm{E}(\varepsilon_{ij}) = \mu. \tag{3}$$

The variance has the form

$$\mathrm{Var}(Y_{ij}) = \mathrm{Var}(\mu + \alpha_i + \varepsilon_{ij}) = \mathrm{Var}(\alpha_i) + \mathrm{Var}(\varepsilon_{ij}) = \sigma_\alpha^2 + \sigma^2. \tag{4}$$

The covariance of two observations of the same group i is

$$\begin{aligned} \mathrm{Cov}(Y_{ij}, Y_{ik}) &= \mathrm{Cov}(\mu + \alpha_i + \varepsilon_{ij}, \mu + \alpha_i + \varepsilon_{ik}) & (5) \\ &= \mathrm{Cov}(\alpha_i, \alpha_i) + \mathrm{Cov}(\alpha_i, \varepsilon_{ij}) + \mathrm{Cov}(\alpha_i, \varepsilon_{ik}) + \mathrm{Cov}(\varepsilon_{ij}, \varepsilon_{ik}) = \mathrm{Var}(\alpha_i) = \sigma_\alpha^2. & (6) \end{aligned}$$

Similarly, we can see that the covariance of observations from two different individuals is zero. Thus, we have

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma^2} \qquad \text{and} \qquad \mathrm{Corr}(Y_{ij}, Y_{lk}) = 0, \quad i \neq l. \tag{7}$$

When fixed and random effect are combined a mixed effects model results. The definition of a linear mixed model is

$$Y_{ij} = \underbrace{\mathbf{X}_{ij}\boldsymbol{\beta}}_{\text{fixed effects}} + \underbrace{\mathbf{Z}_{ij}\boldsymbol{\alpha}_i}_{\text{random effects}} + \varepsilon_{ij}, \tag{8}$$

where $\boldsymbol{\alpha}_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and where $\mathbf{X}_{ij}$ and $\mathbf{Z}_{ij}$ are known matrices of appropriate sizes. Hence, we are also capable of discussing the between-group variability in covariate effects relative to the mean population effect. Note that one can write the model of Equation (8) in vector form as $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$, with appropriate distributional assumptions. Moreover, note that we used generalized mixed effects models in some analyzes, which are an extension of the mixed effects models to non normally distributed response variables; see Bates et al. (2015) for more information.

Estimation in a mixed model is not straightforward because no closed form solutions exist. Three main approaches exist to estimate the parameters of the model: maximum likelihood (ML), restricted maximum likelihood (REML), Markov chain Monte Carlo (MCMC). ML estimation is based on profile likelihood and beneficial to compare different (nested) models but yields less stable variance component estimates than REML. The MCMC approach is computationally demanding.

To fit the models we used the function `lmer()` provided through the package `lme4`. The specification of the model is similar to a fixed effects model. Random effects need to be specified through a `( .| .)` model term. Table A5 gives more practical details, which help to understand the model formulas in Tables A1–A4. A more detailed introduction can be found in Bates et al. (2015) and the references therein.

Table A5: Formula syntax for function `lmer` *in package* `lme4`*. This is helpful to understand the model formulas in Tables A1–A4.*

Formula	Model
`(1 | group)`	random intercept within `group`
`(x | group)`	random slope for `x` and intercept within `group`, with correlation between intercept and slope
`(0 + x | group)`	random slope for `x` within `group`, no variation in intercept
`(1 | group1) + (1 | group2)`	random intercepts for two (crossed or nested) grouping factors
`(x || group)` or `(1 | group) + (0 + x | group)`	uncorrelated random intercept and random slope for `x` within `group`

D The usage of Most Likely Transformations models to relate field measurements with satellite data

As the biodiversity and satellite data considered in this study did no show any correlation, we did not test the Most Likely Transformations (MLT) models with the observed data. Instead, we present a simulation study showing that MLT models are a promising statistical framework to down-scale biodiversity measurements with remotely sensed images. More information about theoretical and practical aspects of MLT models are given in Hothorn et al. (2014, 2015), and Hothorn (2017). We use functions from the R package mlt to fit the models.

In the following, we consider a scenario where satellite observations and punctual biodiversity measurements are available along a one dimensional transect with 1000 locations; see also Figure A16. We generated the data as follows:

(i) the true biodiversity (green line in the lower panel of Figure A16) are simulated from a multivariate Gaussian distribution with a covariance matrix different from the identity matrix.

(ii) the observed biodiversity measurements are the values of the true biodiversity taken at nine locations (red crosses in the lower panel of Figure A16).

(iii) the satellite observations (upper panel of Figure A16) were simulated from a Gaussian distribution such that large biodiversity values were positively correlated with the mean and the variance of the samples. In other words, large biodiversity values implied large satellite observations (in average) and large variability.

The goal of the analysis is to estimate (down-scale) the true and unobserved biodiversity values (green line in Figure A16) from the biodiversity measurements (red crosses in Figure A16) and satellite observations (upper panel of Figure A16). To that end, we first defined nine subsets of the satellite values that were considered to contain valuable information about each of the observed biodiversity measurements. More precisely, the subset corresponding to one biodiversity measurement consisted of all satellite values observed within the interval of 12 values to the left and right of the location of the biodiversity measurement. The subsets are marked with orange in the upper panel of Figure A16. Next, we fitted nine separate MLT models to those subsets describing their distributions with coefficients of an increasing Bernstein basis of order two. Note, that we considered only the subsets of the satellite data for this step and none of the measured biodiversity values.

In order to predict the biodiversity value at a new location with no biodiversity measurement, we first extract the corresponding subset from the satellite data, which includes again all values of the satellite observations within the interval of 12 values to the left and right of the new location. In a second step we calculated the likelihood of the values in that subset for all nine previously fitted MLT models. The biodiversity value belonging to the model with the maximal likelihood value was taken as the predicted value. To obtain predictions for all locations along the transect, we repeated this procedure at all location without observed biodiversity measurements (dashed black line in the lower panel of Figure A16).

An inspection of the fitted biodiversity values reveals that they are generally close to the true unobserved biodiversity values. Moreover, the predicted values are not on a smooth line and, by design, always identical to one of the measured biodiversity values. This also explains why the large and small biodiversity values at the left and right margins, respectively, have relatively bad predictions.

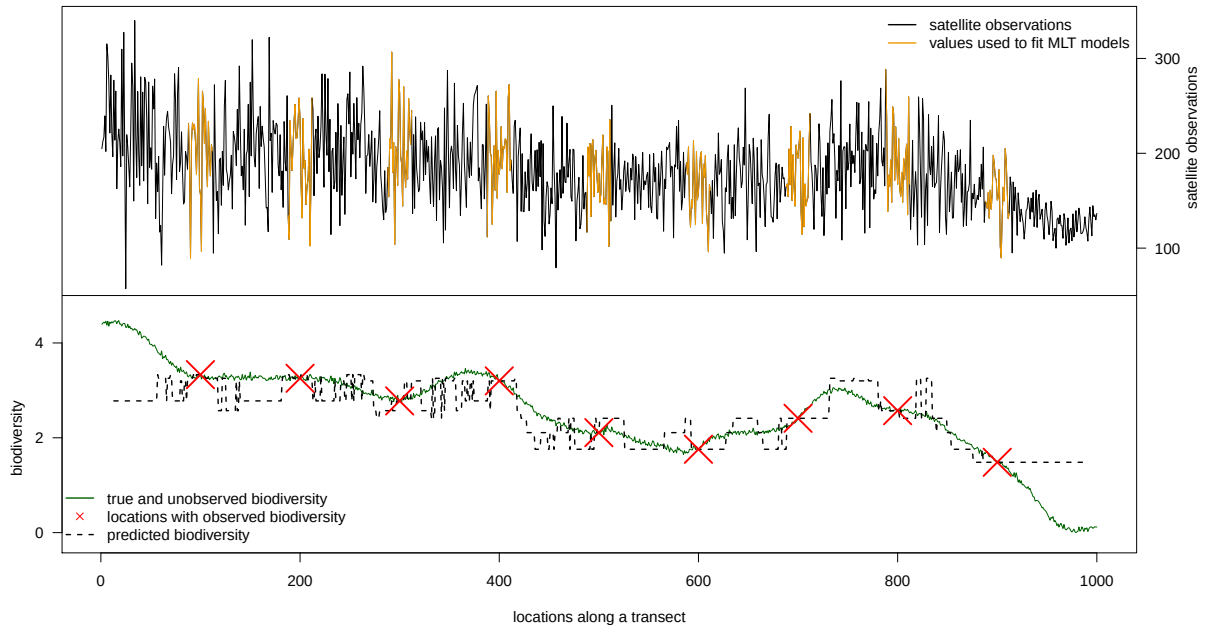


Figure A16: Simulation study showing that MLT models are a promising tool to predict (down-scale) biodiversity estimates from field measurements and satellite observations. The goal is to predict the true biodiversity (green line in the lower panel) at all 1000 *locations along the* x*-axis. Observed are the biodiversity field measurements (red crosses in the lower panel) and the satellite observations (black and orange lines in upper panel). The predicted biodiversity values are shown as dashed black line in the lower panel.*

Acknowledgments

I would like to express my thanks to all the people and institutions that have contributed to the success of my PhD project. Thank you to Prof. Dr. R. Furrer for giving me the opportunity to perform research in the applied statistics group at the Department of Mathematics at the University of Zurich. His guidance contributed much to the success of this work. Moreover, I am very thankful for the flexible working hours and home office opportunities, which allowed me to combine work and family life in a sensible way. Thank you to Dr. G. Schaepman-Strub for co-supervising the PhD thesis. Thank you to Prof. Dr. M. E. Schaepman and Prof. Dr. T. Hothorn for being part of my PhD committee.

Thank you to current and former members of the applied statistics group for the good time: Dr. Z. Baranczuk, Dr. J. Braun, Dr. C. Chevalier, G. Kratzer, Dr. D. Masson, M. Molinaro, Dr. M. Pittavion and C. Wang. Thank you to K. Mösinger and Dr. R. de Jong for contribution to manuscripts of my thesis. Thank you to other colleagues form the Department of Mathematics and the URPP.

Thank you to all of my family. To my mom, dad, and my brothers for the support during my studies. Thank you to Angela for the love and support, and to Elodie for being there. Thank you to my friends. A special mention must also be made to those who have proofread parts of the thesis: Daniel, Nora, Kim, and Anna.

Finally, thank you to the URPP for supporting this position within the Department of Mathematics. I have benefited greatly from this program, and I hope that others will be able to as well in the future.

APPENDIX

Curriculum Vitae

Florian Gerber

Curriculum vitae

Florian Gerber | florian.gerber@math.uzh.ch | July 12, 2017

Address
Name: GERBER
First names: Florian Daniel Markus
Private address: Felsenstrasse 36, CH-4600 Olten
Business address: Department of Mathematics
University of Zurich (UZH)
Winterthurerstrasse 190, CH-8057 Zurich
Cell phone: +41 77 778 45 72
http://www.math.uzh.ch/assistants/flgerb

Personal Data
Date of birth: May 13, 1986
Nationality: Swiss
Heimatort: Langnau i. E. BE
Partner: Angela Pfister
Child: Elodie Pfister, born in 2014

Researcher IDs
http://orcid.org/0000-0001-8545-5263
http://www.researcherid.com/rid/B-7744-2017
https://scholar.google.ch/citations?user=X0nZoXIAAAAJ
https://www.researchgate.net/profile/Florian_Gerber
https://www.scopus.com/authid/detail.uri?authorId=55930181400

Research Interests
- Spatial and spatio-temporal statistics
- Algorithms and software for large datasets
- Bayesian hierarchical models
- Remote sensing based vegetation mapping

Education

08.2013 – present **University of Zurich, PhD candidate**
Department of Mathematics, Applied Statistics Group.
Supervisor: Prof. Dr. R. Furrer. Co-supervisor: Dr. G. Schaepman-Strub.
Zurich Graduate School in Mathematics.

09.2011 – 06.2013 **University of Zurich, MSc in Biostatistics**
Thesis title: Disease mapping with the Besag-York-Mollié model applied to a cancer and a worm infections dataset. Supervisor: Prof. Dr. R. Furrer.

09.2006 – 02.2010 **University of Bern, BSc in Mathematics** (Major) and Philosophy (Minor)
Thesis title: Erweiterungen von Moduln. Supervisor: Prof. Dr. A. Jeanneret.

09.2006 – 06.2008 University of Bern, Bachelor Minor in Interdisciplinary Studies in Ecology.

09.2006 – 02.2007 University of Arts Bern, preparatory course with instrument drums.

07.2002 – 06.2006 Deutsches Gymnasium Biel, Hauptfach: Physik und angewandte Mathematik.

Professional Experience

08.2013 – present **University of Zurich, Institute of Mathematics**
PhD Candidate in the Applied Statistics group of Prof. Dr. R. Furrer
- Research:
  - Prediction of missing values in satellite based vegetation data.
  - Inference with Bayesian hierarchical models.
  - Member of the Research Priority Program on Global Change and Biodiversity.
- Teaching Assistant:
  - Shared responsibility for lectures, exams, homework assignments, and course material.
  - Teach exercise courses for the MSc in Biostatistics and the BSc in Applied Probability and Statistics:
    Einführung in die statistische Modellierung, FS 14; Statistical modeling, HS 14;
    Modeling dependent data, FS 15; Stochastische Modellierung, HS 15;
  - Co-supervision of a MSc thesis on the R implementation of huge covariance matrices.
- Statistical consulting:
  - Methodological support for scientists from the University of Zurich Faculty of Science.

04.2012 – 03.2013 **University of Bern, Institute of Social & Preventive Medicine**
Junior statistician
- Causal inference applied to HIV cohort data.
- Collaboration with the International epidemiological Databases to Evaluate AIDS.

01.2012 – 08.2013 **Novartis Pharma AG, Basel, Statistical Methodology**
External consultant
- Development and maintenance of the R package gsbDesign.
- Methods for the evaluation of Bayesian group sequential clinical trial designs.

01.2011 – 06.2011 **University of Bern, Institute of Social & Preventive Medicine**
Statistical trainee
- Probabilistic record linkage for the Swiss Childhood Cancer Registry (SCCR).
- Development of the R package gsbDesign in collaboration with Novartis Pharma AG, Basel.

01.2009 – 03.2010 Day care Ginkgo in Bern, part time job.

Attended Workshops and Courses

2015, Zurich	International workshop on spatial scaling of Earths biodiversity.
2015, Zurich	Higher-performance R programming via C++ extensions, D. Eddelbuettel.
2014, Zurich	Mixed models with R, Dr. F. Scheipl.
2014, Belgrade	Spatio-temporal modeling of climatic variables using open source software.
2014, Bern	Gaussian random filed simulation workshop.
2013, Wengen	Mathematical modeling in infectious disease epidemiology, Swiss Epidemiology Winter School.

Languages

German (native), English (very good, C1), French (basic A2), Spanish (basic A1).

2010, Ecuador	Spanish training and travel, 6 month.
2005, Canada	English training in British Columbia, 1 month.

Computer Skills

Programming:	R, C++, Python, STATA, Fortan90, MATLAB, Mathematica, MPI, openMP.
OS & Tools:	Linux, BASH shell, Make, Version control with Git and HG Mercurial, Windows, Mac.
Office:	Latex, Sweave/knitr, OfficeLibre, MS Office Suite, GIMP.

Active memberships in scientific societies

- American Statistical Association (ASA)
- Swiss Statistical Society (SSS)

Research output

Peer reviewed articles

Gerber, F., Mösinger, K., and Furrer, R. Extending R packages to support 64-bit compiled code: An illustration with spam64 and GIMMS $NDVI_{3g}$ data. Computers & Geosciences, 104:109–119, 2017. doi: 10.1016/j.cageo.2016.11.015

Gerber, F. and Gsponer, T. gsbDesign: An R package for evaluating the operating characteristics of a group sequential Bayesian design. Journal of Statistical Software, 69(1):1–23, 2016. doi: 10.18637/jss.v069.i11

Bratulic, S., **Gerber, F.**, and Wagner, A. Mistranslation drives the evolution of robustness in tem-1 β-lactamase. Proceedings of the National Academy of Sciences, 112(41):12758–12763, 2015. doi: 10.1073/pnas.1510071112

Gerber, F. and Furrer, R. Pitfalls in the implementation of Bayesian hierarchical modeling of areal count data: An illustration using BYM and Leroux models. Journal of Statistical Software, 63(1):1–32, 2015. doi: 10.18637/jss.v063.c01

Bashir, T., Sailer, C., **Gerber, F.**, Loganathan, N., Bhoopalan, H., Eichenberger, C., Grossniklaus, U., and Ramamurthy, B. Hybridization alters spontaneous mutation rates in a parent-of-origin-dependent fashion in Arabidopsis. Plant Physiology, 165(1):424–437, 2014. doi: 10.1104/pp.114.238451

Gsponer, T., **Gerber, F.**, Bornkamp, B., Ohlssen, D., Vandemeulebroecke, M., and Schmidli, H. A practical guide to Bayesian group sequential designs. Pharmaceutical Statistics, 13(1):71–80, 2014. doi: 10.1002/pst.1593

Gerber, F., Marty, F., Eijkel, G. B., Basler, K., Brunner, E., Furrer, R., and Heeren, R. M. A. Multiorder correction algorithms to remove image distortions from mass spectrometry imaging data sets. Analytical Chemistry, 85(21): 10249–10254, 2013. doi: 10.1021/ac402018e

Wandeler, G., **Gerber, F.**, Rohr, J., Chi, B., Orrell, C., Chimbetete, C., Prozesky, H., Boulle, A., Hoffmann, C., Gsponer, T., Fox, M., Zwahlen, M., Egger, M., and IeDEA S. A. Tenofovir or Zidovudine in second-line antiretroviral therapy after Stavudine failure in Southern Africa. Antiviral Therapy, 19(5):521–525, 2013. doi: 10.3851/imp2710

Articles under review

Gerber, F., de Jong, R., Schaepman, M. E., Schaepman-Strub, G., and Furrer, R. Predicting missing values in spatio-temporal satellite data. IEEE Transactions on Geoscience and Remote Sensing. Submitted, https://arxiv.org/abs/1605.01038

Gerber, F., Mösinger, K., and Furrer, R. dotCall64: An efficient interface to compiled C/C++ and Fortran code supporting long vectors. arXiv, 2017. URL http://arxiv.org/abs/1702.08188

Software

The developed open-source R, C, C++, and Fortran code is available in the following R packages.

Package	Role	URL	CRAN release	Downloads*
spam	Contributor	https://cran.r-project.org/package=spam	2007-09-08	165'933
dotCall64	Contributor	https://cran.r-project.org/package=dotCall64	2016-10-07	263
gapfill	Creator	https://cran.r-project.org/package=gapfill	2016-05-04	619
gsbDesign	Creator	https://cran.r-project.org/package=gsbDesign	2012-05-08	2'974

* Number of downloads in 2016. Only downloads from RStudio users are counted.

Posters at conferences

Gerber, F., Furrer, R., and Schaepman-Strub, G. Linking arctic plant biodiversity measurements with landscape heterogeneity. University of Zurich Reaseach Priority Program Global Change and Biodiversity, 2016. AGU Fall Meeting, San Fracisco, USA

Gerber, F., Furrer, R., and Schaepman-Strub, G. Linking arctic plot scale biodiversity to a remote sensing based landscape characterization. University of Zurich Reaseach Priority Program Global Change and Biodiversity, 2016. Conference on Global Change and Biodiversity: Integrating Mechanisms of Interactions, Feedbacks and Scale, Monte Verità, Ascona, Switzerland

Gerber, F., Furrer, R., Schaepman-Strub, G., and Schaepman, M. Statistical approaches to multiscale biodiversity research. University of Zurich Reaseach Priority Program Global Change and Biodiversity, 2015. Retreat in Asp near Aarau, Switzerland

Gerber, F., Furrer, R., Schaepman-Strub, G., and Schaepman, M. Assessing uncertainty in global change biodiversity research using multiscale Bayesian modeling. University of Zurich Reaseach Priority Program Global Change and Biodiversity, 2014. Retreat in Warth, Switzerland

Gerber, F., Furrer, R., Marty, F., and Brunner, E. Image correction of repetitive distortion patterns in the output of a SIMS experiment. University of Zurich Reaseach Priority Program System Biology & Functional Genomics, 2012. Retreat in Brunnen, Switzerland